Automatic Metadata Discovery from Non-cooperative Digital libraries

Rong Shi
Dept. of Computer Science
Old Dominion University
Norfolk, VA 23529, USA
shi@cs.odu.edu

Kurt Maly
Dept. of Computer Science
Old Dominion University
Norfolk, VA 23529, USA
maly@cs.odu.edu

Mohammad Zubair
Dept. of Computer Science
Old Dominion University
Norfolk, VA 23529, USA
zubair@cs.odu.edu

ABSTRACT

Research into resource discovery for unstructured and structured data on the web has traditionally been conducted as two separate fields. Lately these two areas are converging by each using successful techniques from the other. Building unified higher-level services for a set of heterogeneous digital libraries (structured data) is a challenging task, and particularly when digital libraries are not amenable to modifications (non-cooperating). In this paper, we provide a lightweight, user centered approach based on LFDL (Lightweight Federated Digital Library) and post-processing techniques based on general web search-engines. Specifically, we enhance LFDL by providing a metadata retrieval mechanism from non-cooperating DLs. This allows a user to search seamlessly across a large domain of 'hidden' (from general search engines) web pages and obtain a complete set of metadata for the results (as is normally obtained for results form a single digital library). In addition, this allows metadata to be stored in a local repository to create an intelligent cache for better performance and efficiency.

Keywords

Digital Library; metadata; interoperability; XML; interface; structured data

1. INTRODUCTION

Research into resource discovery for unstructured and structured data on the web has traditionally been conducted as two separate fields. Lately these two areas are converging by each using successful techniques from the other. Building unified higher-level services for a set of heterogeneous digital libraries (structured data) is a challenging task, and particularly when digital libraries are not amenable to modifications (non-cooperating). In the broad scholarly community, a number of free on-line journals and digital libraries exist today, however, sharing metadata among these libraries is difficult. Thus, interoperability is a key area of DL research, facilitated by open frameworks and standards, involving lightweight protocols. General web search-engines have solved the interoperability problem by developing sophisticated crawlers but have significant problems with obtaining results from the 'hidden' web that digital libraries and other proprietary databases inhabit [18]. Also these engines have no way to take advantage of metadata that may be available to characterize web pages (see though efforts with the semantic web [7].

In this paper we concentrate on the approach from the digital library perspective. There are two main approaches that are being pursued to address the interoperability issue: the harvesting approach in which metadata is collected at a central location from different digital libraries to provide higher-lever services such as a unified search interface; and the distributed approach in which metadata continue to reside with original collections and only relevant metadata is retrieved if and when necessary to support a higher-level service. The approach discussed in this paper falls into the later category. In past, researchers have proposed techniques under the distributed approach that either requires modification to the participating digital library or/and installing of some software that works with the participating digital library. The challenges to interoperability here are: (a) the integration should be flexible enough to allow individual participants of the federation to add/modify features and at the same time maintain the user's impression of a single system, and (b) relocation, addition, deletion of individual DLs should be transparent to users.

We addressed some of the issues and challenges in our earlier work in LFDL (Lightweight Federated Digital Library). The resulting testbed system had a fairly high level of quality of service in terms of precision/recall and users were given rich functionalities in their resource discovery. However, not much effort was placed on processing the search results, and they were presented in a flat structure. Organizing of result set helps user to locate the target object quickly in the result set. This requires post-processing of the result set, which is a challenging task in the distributed approach. Recall that the distributed approach in contrast to harvesting approach does not maintain the metadata from different collections locally. Performance is another major issue in a federated centralized service using distributed queries against non-cooperative DLs. In our earlier work on LFDL we improved the performance by using a local cache to store the query results. However, the cache reusability was low as only an exact matched query string resulted in a cache hit. What one needs is a local repository with an "intelligent cache", so that there are more cache hits without reducing the search result quality. Both the tasks, organizing the result set and intelligent caching, require additional processing of the result set using all the metadata available from the result set. However, extracting

WWW 2003, May 20-24, 2003, Budapest, Hungary.

metadata from a DL that is not cooperating is a non-trivial problem[1].

In this paper, we present an automatic metadata discovery and retrieval mechanism based on the same principles we used to provide a search service to non-cooperative DLs: by observing the external behavior of a DL. The DLDL (Digital Library Definition Language) has been enhanced and an XML specification is used to define the rules to obtain metadata from each DL's result pages. Section 2 gives an overview of common approaches in achieving DL interoperability and our earlier work in LFDL. We briefly describe the problem within distributed search, and why it is necessary to do post-processing on result set to obtain metadata. Section 3 gives the details on how we achieve metadata discovery and retrieval from non-cooperative DLs, as well as building a local repository using the metadata harvested. We present some experimentation results in section 4. In the last section we discuss evaluation and future work.

2. BACKGROUND

The NSDL community [13] has been at the forefront of working on federating digital libraries based on early work by [12][6][3][8] and later the Open Archives Initiative [4][5][14][2] and [1][17]. Related work on general web search engines is found in [9] and by the providers such as Lycos, Google, Infoseek, and Vivisimo.

Though many DLs are willing to participate in a interoperable service, still many will continue to work on stand-alone basis, in particular, commercial services. Our emphasis is to achieve interoperability among non-cooperative digital libraries. The nature of totally independent data providers without following any common protocol, by definition prevent building a common service which requires knowledge of each participant's internal structure, language, and protocol. However, by observing each DL's exposed behavior it is still possible to build a such interoperable service, due to the fact that all DLs use an Internet based infrastructure and have universally adopted HTML based web browsers as the common client-side access tool. In our earlier work [15][10][11] on LFDL, we have demonstrated that this is a feasible solution to interoperability among non-cooperative DLs. In LFDL we presented a more general approach to interoperability – Data Centered Interoperability (DCI). We observe the user interaction with the digital library and develop an XML specification of all possible user/DL interactions including the way a DL presents the results of a query to the user.

We created a testbed LFDL implementation in which we defined a universal search interface and a DL definition language to describe a DL's specification. The specification defines the rules of query mapping between universal interface and native interface. The testbed provided a simple search and was successful in demonstrating a lightweight approach to achieve interoperability among non-cooperating DLs. However, the first testbed system had limitation in terms of search capabilities and the quality of service (precision, recall, and performance) was poor. The obstacle to obtain a better precision/recall is that different DLs have very different interfaces with unique features to increase their respective precision/recall for their users. An enhanced LFDL [16] was introduced to provide an interactive, user centered or user-assistant, need-driven advanced search mechanism based on Dublin Core as the basic interoperation middle layer, and a dynamic query mapping mechanism to map between the common layer and the native libraries' layers. The quality of service in terms of precision/recall had been improved and users were given rich functionalities in their searches.

Limitations and Issues

Preliminary tests on the enhanced LFDL system show that providing a federation service for non-cooperating digital libraries is possible and that a dynamic user-centered search interface is a practical approach to improve the quality of service. However, all our work on LFDL and enhanced LFDL were concentrate on tuning the search, we did not concentrate at first on presenting a function-rich, user-friendly search result to end users. From interacting with individual digital libraries users are accustomed to see important information about a result record, such as who the author is, when and where it is published, and what it is really about (abstract, keywords, and/or subject). They may also want to manipulate the results, such as to show only results by a particular author or after a particular date. All these require rich, interactive, and dynamic search result manipulation features. In the earlier work, we could only provide the title and the hyperlink of a result record, grouped by each DL. From the point of view of an end user, such service usability is not satisfactory. Ideally, if we can get all the metadata associated with the records in the search results, we could provide all these services.

Performance is another major issue in a federated centralized service using distributed queries against non-cooperative DLs. A local cache was used in LFDL to improve service performance. All results were cached according to the search query string so that if the same query were submitted, the local cache would be used instead of sending the query to a remote DL. However, such a cache mechanism was not flexible, efficient, and scalable. Cache reusability was low, as only an exact matched query string will result in a cache hit. The cache system would not know which field was a match for a particular query, author or publishing date, for example. Records by author A and records by author A published in year B will have two entries in the cache, which means considerable redundant information and a wasting of resources. Only a search against author A will hit the first entry and only a search against author A and year B will hit the second entry.

3. AUTOMATIC METADATA DISCOVERY AND RETRIEVAL

3.1 Approach

Due to the LFDL's limitations on service usability, scalability, and system performance discussed above, there is need for rich search results and efficient local repository. However, obtaining metadata from a DL without any cooperation is not a trivial task. In our approach a DL does not provide metadata or a way to obtain its metadata; secondly, each DL has its own way to define metadata, and can display any subset of its metadata at its own discretion. The most difficult part is that there is no consistent way among DLs to display metadata; each DL has its own rules as to which metadata to display and in what form.

[1] This is purely a technical contribution and we sidestep the moral and legal issues of making publicly available information available through a third-party service.

Each individual DL provides a search service by three general web-based interfaces: an HTML form-based search page, a list of output pages of search results, and a detail page of a single record/document. In LFDL we use a generic universal search interface based on Dublin Core elements, and we define each DL's behavior by using a specification that is generated based on each DL's search interface. The specification defines the rules of query mapping so that a federated search service can be provided. The results list page and/or document details page provides a possible source of result metadata. Typically, DLs list important meta information about each matched document on the search result page, and the metadata information matches closely the Dublin Core metadata set. Even if no other meta information than the document title and a hyperlink to the document is available on the result page, once a user clicks on the hyperlink more detailed meta information about a particular document or record will be presented to the user. Therefore, an automatic metadata discovery and retrieval from a non-cooperating DL is possible as long as such metadata is available from it's search results page and/or record details page. Our approach is to define rules on how to extract metadata from these pages, and to develop a metadata parser that will use these rules to obtain the metadata.

Handling differences in metadata definition among different DLs is relatively easy. As we defined in LFDL a generic universal search interface, we can use the Dublin Core (DC) metadata set as a common set, and all individual DL's metadata fields are mapped to the closest Dublin Core field. Hence, the LFDL search service will be based on Dublin Core fields. Some DLs may have fields that cannot be mapped to DC fields. We can define a set in addition to DC; if those fields are commonly used, we will map them to the extra set. If a field is unique to a DL, we will still specify it and keep it. The metadata description of a DL will be limited to the exposed fields of that DL.

The difficult part is to define the rules to handle all different cases to gather metadata from different DLs, or how to parse search results and record pages. Ideally, different DLs would use consistent ways to make their metadata publicly available. For example, all DLs could use the <meta> tag to display metadata information on their result and record pages, and moreover, they could all use the same DC element name as <meta> name, then it would be straightforward to define the parsing rules. But unfortunately, in reality each DL has its own way to display such information, and many times no meta tag is used but all information are in the actual HTML code. Therefore, our common metadata retrieval rules have to be generic enough to parse different result pages for different DLs. In Table 1 we list a few sample result pages to illustrate the differences among DLs.

Table1. Sample DL results and metadata display patterns

DL	***Sample result (from results list page)***	***Metadata fields and display pattern***
ACM	*Becoming a computer scientist* Amy Pearl , Martha E. Pollack , Eve Riskin , Elizabeth Wolf , Becky Thomas , Alice Wu **Communications of the ACM** November 1990 Volume 33 Issue 11 It is well known that women are significantly underrepresented in scientific fields in the United States…	title creator1, creator2 publication date description
NEEDS	The Knob & Switch Computer: A Computer Architecture Simulator for Introductory Computer Science (2001) 5 Grant Braught; Computer Science Teaching Center Last Updated: 2002-02-01, Score: 336	title creator1, creator2; affiliation date, score
CogPrints	Barlow, Horace (1996) Intraneuronal information processing, directional selectivity and memory for spatio-temporal sequences.. *Network: Computation in Neural Systems* 7:251-259.	creator (date) title publication
CSTC	Integrating Empirical Methods into Computer Science Author: David Reed (davereed@creighton.edu) Date: 05-05-2002 Category: Reviewed Demonstrations from Conferences Subject: Software - Programming Techniques	title creater1, creater2; date category subject
LTRS	1562 50 HZETRN: Description of a Free-Space Ion and Nucleon Transport and Shielding Comp	title
NACA	885 27 Central automatic data processing system	title
WCR	Analysis and modeling of World Wide Web traffic Conference Paper – G. Abdulla – Dept. of Computer Science, Va Polytechnic Institute and State University -- 1998	title type -- creator1, creator2 -- affiliation -- date

3.2 Architecture

To realize the approach described in section 3.1, we start with the overall LFDL system architecture design that is shown in the Figure 1. The basic idea is to add DL metadata parsing rules to the DL's specification and to have the LFDL use these rules to parse the DL's result pages. All parsed metadata is stored in a local repository so that they it can be reused by future search queries.

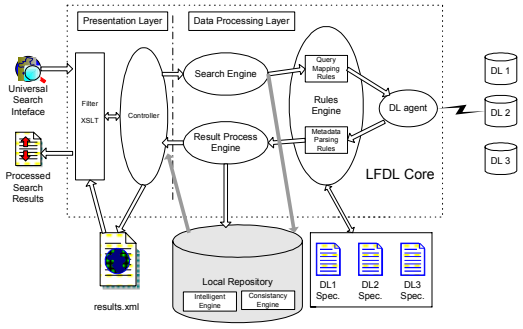

Figure1. LFDL Architecture and Data Flow

The data flow and interaction among the modules is as follows:

- At initialization the system reads all DL specifications including query mapping rules and metadata parsing rules
- A resource discovery user submits a query using the universal search interface
- The front-end filter does pre-processing (query clean-up) with the help of a central controller, and then the query is passed to the Search Engine
- The Search Engine uses the query mapping rules to transform the universal query to a DL's native local query
- A DL agent sends the transformed query to the remote DL and receives the search results
- The Result Process Engine parses the search results pages and extract the metadata according to the metadata parsing rules and store them in the Local Repository
- All parsed results are merged by the Controller into an intermediate XML document
- The resulting XML document is displayed using a XSLT processor.
- Once the Local Repository has been populated, the Search Engine executes searches against the Local Repository (cache) first instead of sending queries directly to remote DLs.

3.3 Metadata Retrieval and Parsing

As we stated in section 3.1, we define DL output metadata at two levels: results list page level, and if available, record page level. Still, some DLs do not provide any metadata at all. Following is the algorithm used by the Result Process Engine to retrieve and parse metadata from HTML pages in two levels.

1. Once search results from a DL arrive, the Result Process Engine checks for parsing rules from the DL's specification.
2. The Process Engine applies parsing rules to get metadata from HTML page, and the generated metadata will be stored in a cache.
3. If DL specification also defines lower level (record page level) metadata parsing rules, all record HTML pages will be retrieved from remote DL, and parsed
4. Extra process on cached metadata so that they are ready to be displayed.
5. After post-processing is done for all results from all DLs, results are merged and then displayed to end-users.
6. Periodically, cached metadata will be saved to persistent storage such as a database.

3.4 Metadata Parsing Rules Definition

We use the same DL XML specification to define metadata-parsing rules as we use for query mapping and metadata retrieval. We extend the DLDL to define parsing rules at two page levels: result list page level and single record document level. As shown in the DTD, see Figure 2, the basic idea is that the raw string is separated into several segments, each segment has one or several metadata fields. MATCH-START and MATCH-END specify a segment, and EXCLUDE and REPLACE will remove unrelated strings. Actual metadata fields will be separated by DELIMETER.

```
<!ELEMENT RESULT-METADATA (MATCH-START,MATCH-
END,EXCLUDE*,REPLACE*,DELIMETER*,METADATA-FIELD*)>
<!ELEMENT RECORD-METADATA (MATCH-START?,MATCH-
END?,EXCLUDE*,REPLACE*,DELIMETER*,METADATA-FIELD*)>
<!ELEMENT METADATA-FIELD (#PCDATA)>
<!ATTLIST METADATA-FIELD
    Title CDATA "information about a particular metadata field">
<!ATTLIST METADATA-FIELD
    order CDATA #IMPLIED>
<!ATTLIST METADATA-FIELD
    multiple (true | false) #IMPLIED>
<!ATTLIST METADATA-FIELD
    delimeter CDATA #IMPLIED>
<!ATTLIST METADATA-FIELD
    format CDATA #IMPLIED>
<!ATTLIST METADATA-FIELD
    null_value_string CDATA #IMPLIED>
```

Figure 2. Part of DTD for DL parsing rule specification

3.5 Local Repository

Once metadata are parsed, they are stored in a local database to form a repository so that all future searches will be checked locally first before sending queries out to remote DLs. By using such a local repository, both search performance and service reliability will be improved. We call this "intelligence cache" as compared with the old caching mechanism in the earlier versions of LFDL. By using a cache grouped by metadata fields, we can provide service at a quality as good as or close to the search service provided by an individual DL that maintains all the data it serves. A consistency engine will handle the cache consistency between local storage and remote DLs.

4. RESULTS

We have implemented this architecture and created specifications for seven digital libraries (ACM, NEEDS, NACA, COGPRINTS, CSTC, LTRS, and WCR). All of these libraries are from the federation of the LFDL, hence we only had to add the parsing and extraction rules to the LD specification documents. We illustrate the process for both a list page level DL and a record level DL. Figures 3 and 4 show the form the metadata two very different DLs present them to the user. The ACM DL in Figure 3 displays considerable amount of metadata information on the list page result and we give in Figure 5 a part of the specification that guides our engine in the extraction process. However, Cogprints displays on the list page results only the title metadata and the user has to click the title to obtain a record level page. Only that page has the metadata of interest. Again, Figure 6 shows the XML specification for the extraction process. From the experience of adding the seven DLs to our federation we can say that n the average the effort to observe and analyze anew DL is on the order of hours rather than days; these specific DLs took an average of three hours to define. This bodes well for the scalability of the approach at least from the specification perspective.

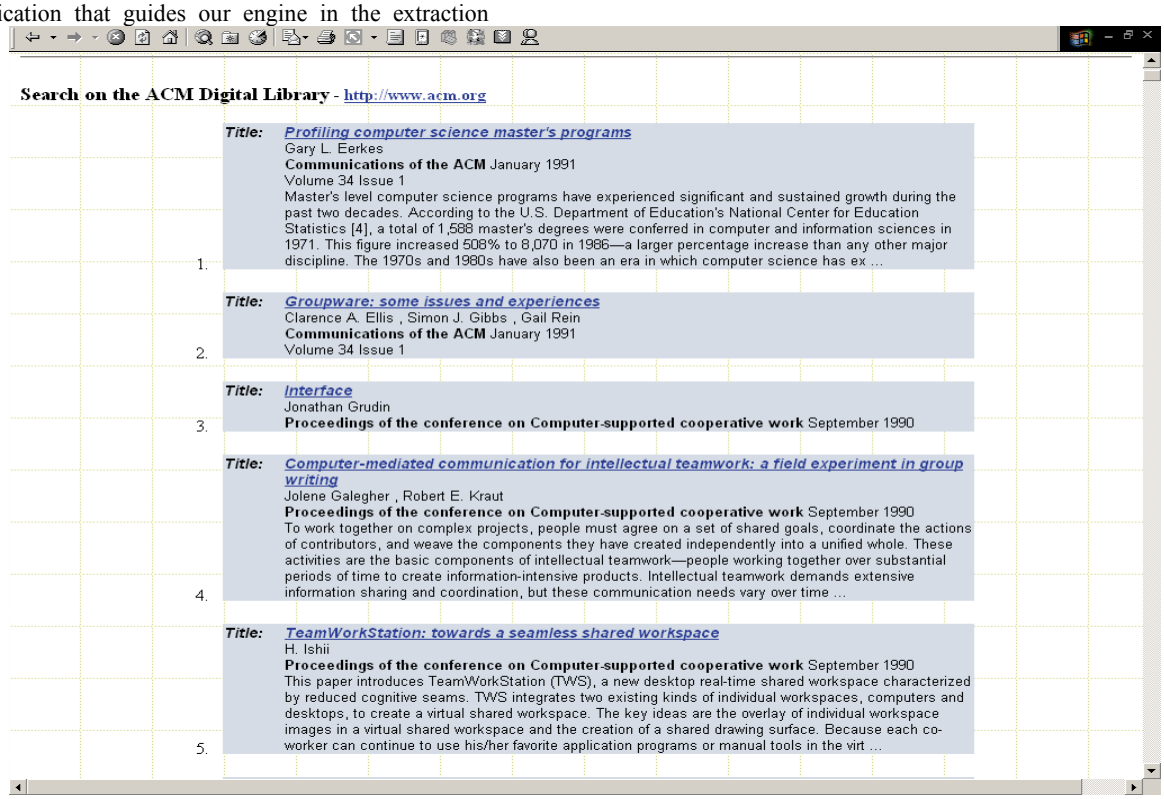

Figure 3. Sample Search Results of ACM DL

Figure 4. Sample Results List Page and Record Page of Cogprints DL

```
- <RESULT-METADATA Title="Result page metadata parsing:" hasRecordLevel="false">
    <MATCH-START enforced="true" Title="the beginning of matching string of result metadata"><div class="authors"></MATCH-START>
    <MATCH-END enforced="true" Title="the end of matching string of result metadata"></div></MATCH-END>
    <EXCLUDE Title="the string should be excluded or removed when parsing"></EXCLUDE>
    <EXCLUDE Title="the string should be excluded or removed when parsing"></EXCLUDE>
    <EXCLUDE Title="the string should be excluded or removed when parsing"></EXCLUDE>
    <METADATA-FIELD order="1" multiple="true" delimeter="," Title="information about a particular metadata field">CREATOR</METADATA-FIELD>
</RESULT-METADATA>
```

Figure 5. Part of DL Specification for ACM

```
- <RESULT-METADATA Title="Result page metadata parsing:" hasRecordLevel="true">
    <MATCH-START Title="the beginning of matching string of result metadata">null</MATCH-START>
    <MATCH-END Title="the end of matching string of result metadata">null</MATCH-END>
  </RESULT-METADATA>
- <RECORD-METADATA Title="Record page metadata parsing:">
    <MATCH-START Title="the beginning of matching string of result metadata">name="DC.title"</MATCH-START>
    <MATCH-END isLastIndex="true" Title="the end of matching string of result metadata">" name="DC.creator"</MATCH-END>
    <EXCLUDE Title="the string should be excluded or removed when parsing">/><meta content="</EXCLUDE>
  - <REPLACE Title="replace old string with new string">
      <OLD-STRING Title="the old string to be replaced">" name="DC.creator"</OLD-STRING>
      <NEW-STRING Title="replace with the new string">;</NEW-STRING>
    </REPLACE>
    <METADATA-FIELD order="1" multiple="true" delimeter=";" Title="information about a particular metadata field">CREATOR</METADATA-FIELD>
</RECORD-METADATA>
```

Figure 6. Part of DL Specification for Cogprints

Finally, Figure 7 shows the results of our LFDL with metadata extraction. Both ACM and Cogprints appear as part of the LFDL results set in the same format and with metadata singled out. The amount of metadata will differ for each library and depends naturally on how much a library exposes in the result set.

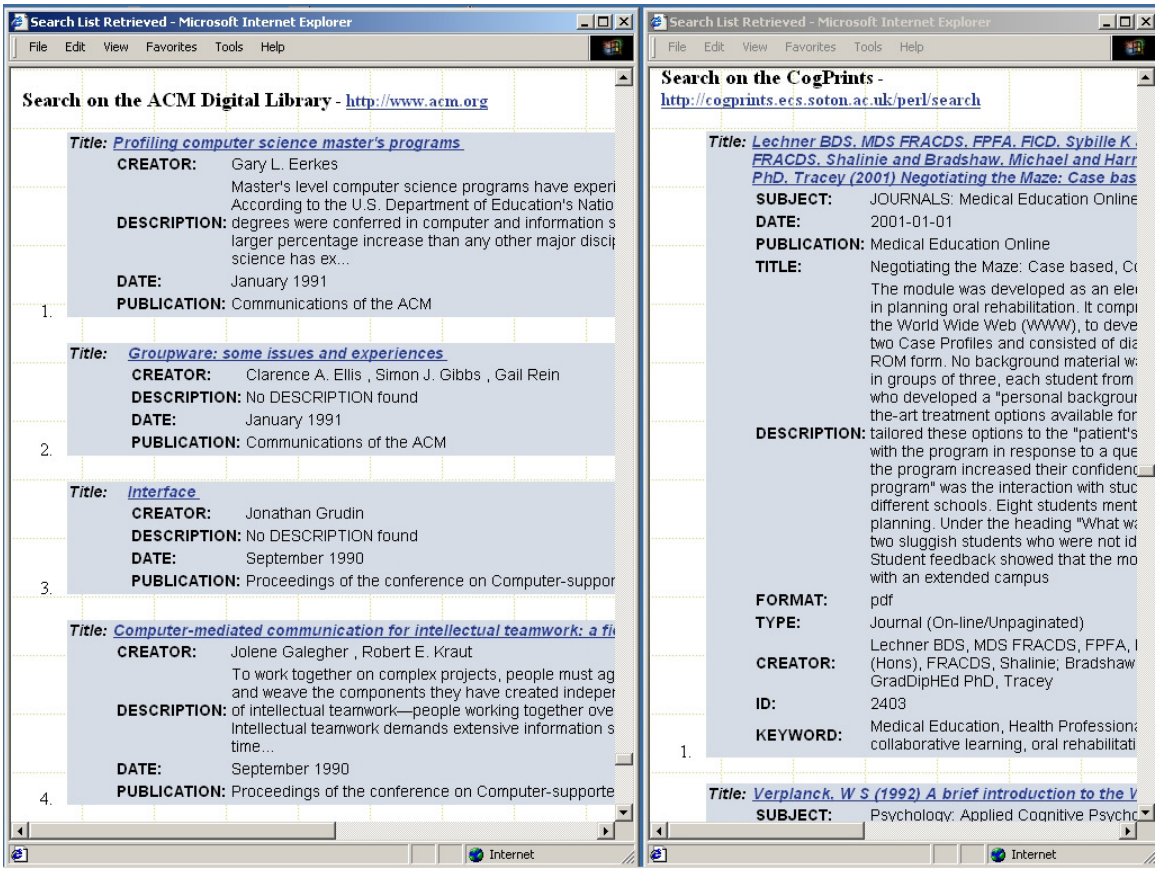

Figure 7. Post Processed Results in LFDL after Metadata Parsing

5. FUTURE WORKS

We have demonstrated the automatic extraction of metadata from non-cooperating digital libraries by implementing successfully a testbed containing seven DLs. We are now working on how to utilize the extracted metadata both to improve the performance through intelligent caching and to increase the quality of service to the user through better result set navigation tools. We are designing a rich, user-friendly, and flexible result-displaying mechanism, so that users can group or process results by any metadata field, without re-sending the query or start a new search. The major direction in this effort will come through focus groups and usability testing of the existing LFDL. The intelligent caching problem we have outlined in the paper and that has been enabled in our architecture needs considerable more design for us to achieve efficiency, flexibility and usefulness. The first problem is to populate the cache. Though we already have a basic keyword set, see [16] and could use them to populate the cache, such process is very time consuming and produces redundant information. Similarly, to populate the cache through real user searches will take a long time before a reasonable size cache can be created and thus be useful to searchers. We will investigate trade-offs and other approaches. How to maintain the cache is another problem; for instance, what size is best considering resource efficiency and cache usage? How do we keep the cache consistent with remote DLs? A third problem concerns the intelligent caching of compound queries, typical query optimizations do not pose queries to a database when the first part of an 'and' query fails. Do we take into account such query optimizations for caching elements of compound queries?

In the meantime we encourage experimentation with our system as we shall shortly make it available as an Open Source system.

6. REFERENCES

[1] A. Paepcke, "Search Middleware and the Simple Digital Library Interoperability Protocol", D-Lib Magazine, March 2000, vol. 6 No. 3, http://www.dlib.org/dlib/march00/paepcke/03paepcke.html

[2] Arc. http://arc.cs.odu.edu

[3] B. M. Leiner, "The NCSTRL Approach to Open Architecture for the Confederated Digital Library", D-Lib Magazine, December 1998. http://www.dlib.org/dlib/december98/leiner/12leiner.html.

[4] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, M. F. Schwartz, and D. P. Wessels, "Harvest: A Scalable, Customizable Discovery and Access System", University of Colorado Computer Science Technical Report CU-CS-732-94, August 1994

[5] H. V. de Sompel, C. Lagoze, "The Open Archives Initiative Protocol for Metadata Harvesting", January, 2001, http://www.openarchives.org/OAI/openarchivesprotocol.htm

[6] J. R. Davis, D. B. Krafft, and C. Lagoze, “Dienst: Building a Production Technical Report Server", Advances in Digital Libraries, Springer-Verlag, 1995, pp. 211-222

[7] Kartoo, www.kartoo.com

[8] K. Maly, M. L. Nelson, and M. Zubair, “Smart Objects, Dumb Archives - A User-Centric, Layered Digital Library Framework”, D-Lib Magazine, March 1999. http://www.dlib.org/dlib/march99/maly/03maly.html.

[9] M. Koster, “The Web Robots Page”, available at http://info.webcrawler.com/mak/projects/robots/robots.html

[10] M. Zubair, K. Maly and I. Ameerally, M. Nelson, “Dynamic Construction of Federated Digital Libraries”, WWW9, pp56-57, Amsterdam, May 2000

[11] M. Zubair, K.Maly, and R. Shi, “Focus Research Libraries in Support of Active Learning”, ED-ICT2000, Vienna, Dec 2000

[12] M. L. Nelson, K. Maly, S. N. T. Shen, and M. Zubair. NCSTRL+: Adding multi-discipline and multi-genre support to the dienst protocol using clusters and buckets. In Proceedings of IEEE Advances in Digital Libraries 98, pages 128–136, Apr. 1998

[13] NSDL, National SMETE Digital Library. http://www.smete.org/nsdl/

[14] OAi. http://www.openarchives.org/

[15] R. Shi, K. Maly and M. Zubair, “Interoperable Federated Digital Library using XML and LDAP”, Global Digital Library Development in the New Millennium, pages 277-286, May 2001

[16] R. Shi, K. Maly and M. Zubair, “Dynamic Interoperation of Non-cooperating Digital Libraries”, DLOC2002, International Conference on Digital Library --- IT Opportunities and Challenges in the New Millennium, Beijing, China, July, 2002

[17] Sergey Melnik et al., “Generic Interoperability Framework”, working paper, http://www-diglib.stanford.edu/diglib/ginf/WD/ginf-overview/

[18] X. Liu, K. Maly and M. Zubair, "DP9- An OAI Gateway Service for Web Crawlers", JCDL2002, pp. 283-284, Portland, Oregon, July 2002