Extending SDARTS: Extracting Metadata from Web Databases and Interfacing with the Open Archives Initiative

Implementation paper

Panagiotis G. Ipeirotis
pirot@cs.columbia.edu

Tom Barry
tjbarry@earthlink.net

Luis Gravano
gravano@cs.columbia.edu

Computer Science Dept.
Columbia University

ABSTRACT

SDARTS is a protocol and toolkit designed to facilitate metasearching. SDARTS combines two complementary existing protocols, SDLIP and STARTS, to define a uniform interface that collections should support for searching and exporting metasearch-related metadata. SDARTS also includes a toolkit with wrappers that are easily customized to make both local and remote document collections SDARTS-compliant. This paper describes two significant ways in which we have extended the SDARTS toolkit. First, we have added a tool that automatically builds rich content summaries for remote web collections by probing the collections with appropriate queries. These content summaries can then be used by a metasearcher to select over which collections to evaluate a given query. Second, we have enhanced the SDARTS toolkit so that all SDARTS-compliant collections export their metadata under the emerging Open Archives Initiative (OAI) protocol. Conversely, the SDARTS toolkit now also allows all OAI-compliant collections to be made SDARTS-compliant with minimal effort. As a result, we implemented a bridge between SDARTS and OAI, which will facilitate easy interoperability among a potentially large number of collections. The SDARTS toolkit, with all related documentation and source code, is publicly available at `http://sdarts.cs.columbia.edu`.

1. INTRODUCTION

A significant amount of valuable information on the web is stored in databases, some of which is "hidden" behind search interfaces and not crawlable by traditional search engines. An attractive way to allow easy interaction with these databases is through *metasearchers* (e.g., [19, 7]), which provide users with a single interface to query multiple databases simultaneously. A metasearcher performs three main tasks: After receiving a query, it determines the best databases to evaluate the query (*database selection*), it translates the query in a suitable form for each database (*query translation*), and finally it retrieves and merges the results from the different databases (*result merging*) and returns them to the user using a uniform interface. These tasks are challenging, which makes it difficult to build a sophisticated metasearcher. Matters are complicated even further by the lack of a common way to interact with the different databases.

To facilitate metasearching, we defined a protocol and an associated toolkit, named SDARTS [10], which provides the necessary infrastructure for incorporating new collections to our NSF Digital Library Initiative–Phase 2 project PERSIVAL [13][1] with minimal effort. SDARTS is a hybrid between two existing protocols, SDLIP [18] and STARTS [8], and inherits from the latter the specification of the metadata that collections should export to make metasearching easier. Collections have to indicate, for example, what attributes are available for searching (e.g., "author"). This information is helpful for the query translation metasearch task. Additionally, collections should export a *content summary* to assist in database selection. This content summary of a collection includes the vocabulary of the collection plus simple statistics like the frequency of occurrence of each word in the collection. The content summaries encompass the core of the information content needed by most database selection techniques that have been proposed in the literature [20, 9, 14, 22].

SDARTS, as reported in [10], supported three types of wrappers: for local text collections, for local XML collections, and for remote web collections. The SDARTS toolkit could construct content summaries for the first two types of collections, to which the SDARTS wrappers have complete access. (These collections are locally available.) Unfortunately, the SDARTS toolkit did not provide any support to build content summaries for the third type of collection, in which documents might not be available other than via a web-accessible search interface. In this paper we describe how we extended the SDARTS toolkit with a method for automatic extraction of content summaries from autonomous, remote databases that do not export any metadata about their contents. By adding this functionality to SDARTS we

[1] `http://persival.cs.columbia.edu`

can have uniform metadata for all SDARTS-compliant collections, no matter if we have complete access to their full contents or not.

To allow interoperability, the SDARTS metadata is exported using the metadata interface of SDLIP, an open Digital Libraries interoperability protocol, developed mainly by digital libraries projects in California. Of course, there are other protocols for exporting metadata like the Open Archives Initiative (OAI) [17], a notable example that is getting momentum among Digital Libraries projects and information providers. Since interoperability is a key goal behind the development of SDARTS, in this paper we also report how we export the SDARTS metadata under the OAI protocol. As a result, all SDARTS-compliant collections become immediately part of OAI. Conversely, we added to SDARTS the capability to harvest and index OAI collections, enriching them with additional search functionality and metadata to make the OAI collections fully SDARTS compliant. As a result, all OAI collections can become SDARTS-compliant with minimal effort. The SDARTS-generated content summaries for OAI collections can then be exploited by metasearchers for efficient distributed search over OAI collections. By examining the content summaries of different OAI collections, it is possible to find the most promising collections and harvest metadata only from them.

The rest of the paper is organized as follows: In Section 2 we briefly review the SDARTS protocol and its associated toolkit. Then, in Section 3 we describe how we built into SDARTS the capability of automatically extracting content summaries from web databases. In Section 4 we discuss how we extended the SDARTS toolkit so that we can easily make OAI collections SDARTS-compliant and vice versa. Finally, in Section 5 we conclude and summarize our contributions.

2. OVERVIEW OF SDARTS

Metasearching is an important component of our PERSIVAL Digital Library project. We have to access different types of collections for users with different needs. The only way to interact with many "remote" collections is through querying, but broadcasting queries to all available collections is neither efficient nor effective. Metasearching can be greatly facilitated if all the databases export a uniform interface for searching, returning results, and exporting metadata about their contents. To specify such an interface, we developed SDARTS [10], a protocol and toolkit designed to help metasearching. To allow interoperability, SDARTS is based on *existing* protocols, namely on SDLIP [18] and STARTS [8], and combines them to create a powerful protocol, which helps make distributed search easier and more effective. We briefly discuss STARTS and SDLIP in Section 2.1 and in Section 2.2 we review the SDARTS protocol and its associated toolkit, which were introduced in [10].

2.1 STARTS and SDLIP

STARTS [8] is a protocol proposal that defines the information that a collection should export to facilitate metasearching. A key piece of information that a collection should export under STARTS is its *content summary*. The content summary of a collection is aggregated information about the contents of the entire collection and contains simple statistics like the frequency of the words that appear in the collection. Figure 1 shows part of the content summary of the "20 Newsgroups" collection [2], a collection that is frequently used in the machine learning community. This collection consists of 19,997 articles posted to 20 newsgroups. 34 of these articles contain the word "algorithm" in their body, as the content summary in the figure indicates. Also, this word appears 75 times over these 34 articles. A metasearcher can use these content summaries to select the collections to send a given query (i.e., for database selection). Most database selection algorithms (e.g., [20, 9, 14, 22]) depend on this kind of content summaries. More specifically, the database selection component of a metasearcher can use this information to estimate the number of relevant documents that a given query will retrieve from each collection. As a simple example, given a query on "algorithm" a metasearcher might conclude that it does not need to query a collection with few or no documents containing that word, as reported in the collection's content summary.

```
<starts:scontent-summary
xmlns:starts=http://sdarts.cs.columbia.edu/STARTS/
version="Starts 1.0"
stemming="false" stopwords="false"
case-sensitive="true" fields="false"
numdocs="19997">
...
<starts:term>
<starts:value>algorithm</starts:value>
</starts:term>
<starts:term-freq>75</starts:term-freq>
<starts:doc-freq>34</starts:doc-freq>
...
```

Figure 1: A small fraction of the STARTS content summary for the "20 Newsgroups" collection.

SDLIP [18] is a layered protocol that defines simple interfaces for interoperability between heterogeneous databases. Its designers define it as "search middleware," lighter and easier to use for web-related applications than standard middleware protocols like Z39.50 [1]. The main purpose of SDLIP is to provide uniform interfaces to collections for querying and retrieving results, and taking care of the transport of the data across the network. SDLIP defines the interfaces that a database or a wrapper of a database should implement so that it can be accessed by an SDLIP-enabled client. An implementation of the SDLIP interfaces functions as a lower-level wrapper around one or more underlying collections. These collections can either be hosted locally, or be remote databases that are accessed over the network. Figure 2, borrowed from [18], depicts the role of SDLIP in a digital library architecture. Essentially, SDLIP masks the heterogeneities of the different search interfaces and exports a uniform interface for all of its underlying collections. SDLIP is designed for clients that know which collection they wish to access. In contrast, STARTS specifies, among other things, the metadata that databases should export to make it easier for a metasearcher to decide which databases it is worth contacting. Hence, by combining the two protocols we created a protocol that allows easy interaction with different collections, and also exports the necessary metadata information to make metasearching more efficient and effective. The protocol that resulted from combining SDLIP

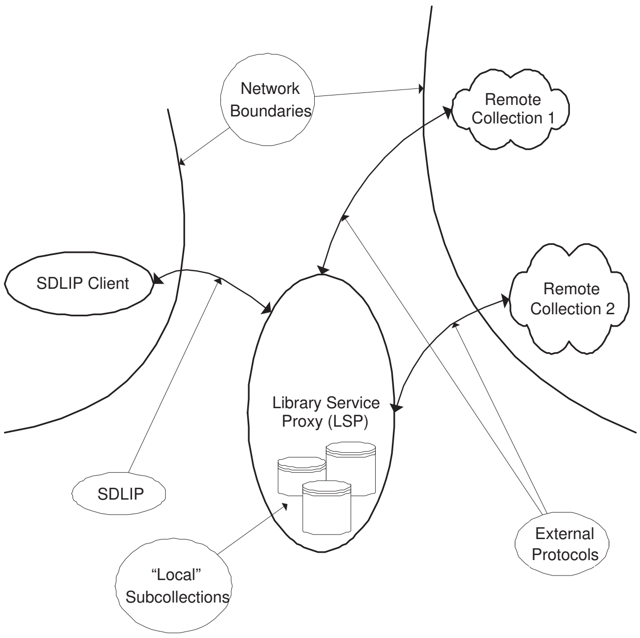

Figure 2: The role of SDLIP in a digital library architecture with autonomous databases and wrappers (taken from [18]).

and STARTS is named SDARTS [10] and we review it next.

2.2 SDARTS

SDARTS was designed with two goals in mind: interoperability and ease of use.

Interoperability is especially important in a digital library environment, where there is a crucial need to access a large number of diverse resources. By using SDLIP, an *existing* protocol, we ensured that we can interoperate with other Digital Libraries projects that had developed and used this protocol for interoperability. We standardized on STARTS as the XML "dialect" for SDLIP to exchange the extra information specified by STARTS and not included in the original version of SDLIP. SDARTS follows all of SDLIP's original DTDs, which include placeholders that can be exploited to include the necessary STARTS XML objects. By standardizing on one XML format for SDLIP, we have created an architecture that allows easy implementation of wrappers for different types of collections (for details, see [10]).

Although the SDARTS architecture makes it easy to implement new wrappers from the developer's point of view, we wanted to create a tool that would allow end-users to wrap the most common types of existing text-based collections with minimal effort. For this reason, the SDARTS toolkit includes three reference wrappers that can be used to cover the vast majority of textual collections. Specifically, we have created one wrapper for simple "local" text document collections, one wrapper for "local" XML-formatted document collections, and one wrapper for "remote" web-accessible collections. All these reference wrappers can be configured easily. We briefly describe the three reference wrappers below:

Text wrappers: This wrapper is designed to make a local collection of text documents fully searchable and available under SDARTS. The SDARTS administrator just needs to write a small configuration file that specifies the local path to the set of documents. Optionally, it is possible to specify a set of stop words, and write regular expressions to locate different fields (e.g., title and author) that exist in the text files. After writing the configuration file, the SDARTS wrapper indexes the files and provides a SDARTS query and metadata interface for the collection. At the same time the content summary of the collection is created and made accessible through the metadata interface. We stress here that the collection metadata is extracted automatically during indexing and made available after the configuration of the wrapper.

XML wrappers: This wrapper is designed to make a local collection of XML-formatted documents fully searchable and available under SDARTS. This wrapper is quite similar to its plain-text counterpart; the key difference is that, to index collections of XML documents, a slightly more complicated configuration file is needed. Except for the information about where the documents are stored, an XSLT stylesheet is needed to define how to find the fields in the documents during indexing. Again, after the initial configuration, indexing and searching are completely automatic. Since the full contents of the collection are always available for inspection, detailed metadata is created automatically and exported using the SDLIP-specified interfaces.

Web wrappers: This wrapper is designed to make a collection hosted in a remote web server fully searchable and available under SDARTS. This is the most complex of the three wrappers. It is intended for autonomous, remote collections fronted by HTML forms and CGI scripts, such as the PubMed collection of medical bibliography [2] and many others. We decided that the best way to make this wrapper configurable without additional Java coding was through XSLT stylesheets. The XSLT stylesheet in this case has an element to describe CGI invocations. This consists of a URL, the query method (GET or POST), and a set of name/value pairs that are the script's parameters. To process a query, the wrapper sends it to the remote engine and gets back the results in HTML format. Then, the wrapper converts the HTML into well-formed XML and this XML is then transformed using an XSLT stylesheet to find the fields of interest in the results page. Thus the SDARTS search interface can be supported by the wrapper. In contrast, exporting content summaries presents complications if the whole contents of a remote collection are not available to the SDARTS wrapper, as is often the case. Hence, it might not be possible to generate metadata about the individual records of the collection in the way we do for the text and XML wrappers.

In summary, the wrappers provided in the SDARTS toolkit allow us to index, search, and extract metadata from every plain-text or XML-formatted collection that is available locally. However, for remote collections over which we have no

[2] `http://www.ncbi.nlm.nih.gov/PubMed/`

control, content summary extraction is challenging. Next, in Section 3, we describe how to automate content-summary generation for such "uncooperative" web databases.

3. EXTRACTING CONTENT SUMMARIES FROM WEB DATABASES

The SDARTS text and XML wrappers extract complete metadata from locally available text and XML collections. In contrast, a web wrapper cannot generate content summaries for remote collections. In this section, we describe how we extended the SDARTS toolkit to address this limitation of web wrappers. In particular, in Section 3.1 we present a brief overview of our content-summary extraction algorithm of choice, while in Section 3.2 we discuss how we implemented this algorithm into SDARTS.

3.1 Content Summary Extraction via Query Probing

To create a complete SDARTS content summary for a web database, we would have to somehow inspect the whole contents of the database. This might not only be impractical but even infeasible, since many collections might be too large to crawl or even uncrawlable altogether (e.g., when their contents are not HTML documents). To circumvent these problems and still be able to automatically build good quality content summaries, we resort to *document sampling*.

A good-quality content summary of a collection can be derived from a small, representative document sample from the collection [3]. Earlier research has shown that we can extract such a document sample with a relatively small number of *query probes* [3, 4, 11]. An *approximate* content summary can then be built from the documents that best match each query probe at the collection in question. Interestingly, the effectiveness of the best database selection algorithms does not suffer significantly from using approximate content summaries extracted in this way, rather than the "complete" content summaries for which the algorithms were originally designed [6, 4].

To make the generation of content summaries in SDARTS completely automatic we adopted the method we described in [11], which uses a small number of topically-focused query probes to produce highly accurate approximate content summaries. Our probing method relies on short query probes generated from document classifiers and has two steps: a training step and a sampling step. In the training step, we start with a comprehensive, predefined topic hierarchy with an associated training set of preclassified documents. Then, we train a rule-based document classifier [5] to produce rules like "IF lung AND cancer THEN Health". According to this rule, a document having the words "cancer" and "lung" will be classified into category "Health". In the sampling step, we transform each of these rules into a query probe (a query containing all the words in the antecedent of the given rule), and issue the queries to the collections for which we want to create the content summaries. We retrieve the top-k documents returned by each query. (A good value for k is four.) The first round of query probes is associated with the top level of our classification hierarchy. Therefore, these queries cover a wide variety of "general" topics. Our algorithm inspects the number of documents that match each query in the collection and decides on which topics to focus. The algorithm then issues a second round of query probes corresponding to the subcategories of the most representative top-level categories for the collection, ignoring other off-topic categories. After all levels in the categorization scheme have been processed in this way, the algorithm returns the top-k documents retrieved for each query probe as the document sample for the collection.

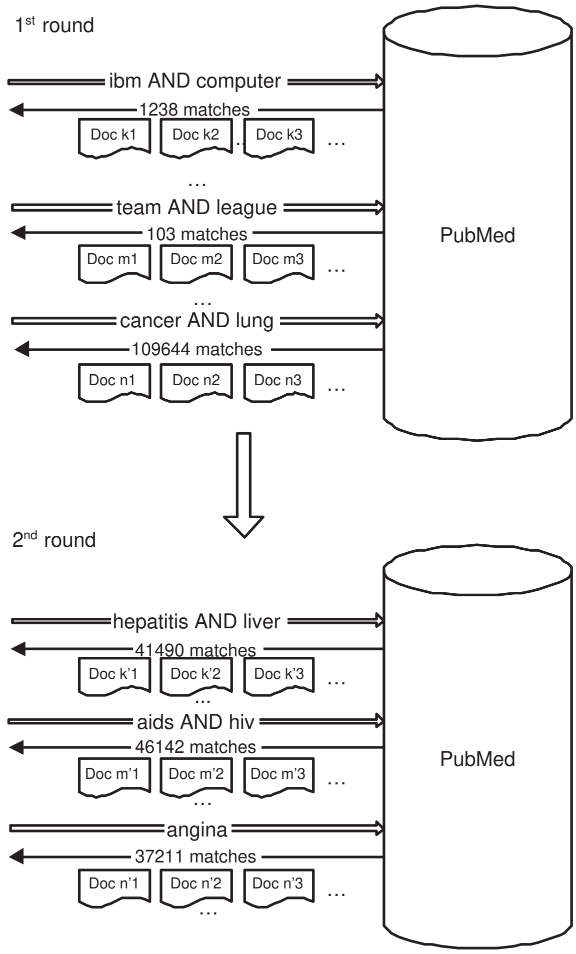

Figure 3: Performing topic-based document sampling on the web-accessible database PubMed.

Figure 3 illustrates how the probing process works. The PubMed database is initially probed with "general" queries, associated with the top-level categories of a classification scheme (e.g., "Health," "Sports"). As a result, we extract a small document sample, which is saved. By examining the generated number of matches for each probe, the algorithm detects that a significant percentage of the documents in the database are about "Health." Hence, it starts a new round of sampling by sending query probes associated with the subcategories of "Health" and ignoring, for example, the "Sports" subtree. The process continues, and the result is a document sample that is representative of the collection contents.

Word	*Document Frequency*
cancer	1,122,844
bone	385,128
hepatitis	80,375
...	...
java	2,393
hunting	912
basketball	880

Table 1: A small fraction of the automatically extracted content summary for the PubMed collection.

After probing a collection as above, we have a document sample that is representative of the collection contents. By inspecting this sample, we can order all words according to the number of sample documents in which they occur. Furthermore, we know the *real* document frequency in the entire collection for those words that appeared as single-word query probes. The number of matches reported for such queries are (a close approximation to) the actual document frequency of the corresponding words. Hence we can use curve-fitting techniques to predict the real document frequency of those words in the document sample that did not appear in query probes by themselves. The result of this process, which is presented in detail in [11], is a content summary that accurately reflects the contents and size of the web collection.

Table 1 reports a fraction of the content summary that we automatically generated for the PubMed collection. We can see that high-frequency words like "cancer" are representative of the topic coverage of PubMed, unlike low-frequency words like "basketball." We created the PubMed content summary by sending less than 200 queries to PubMed and retrieving four documents per query. Also the queries sent to the PubMed search interface are quite short. More than half of the queries consist of only one word, with a three-word maximum. This example illustrates how we can create accurate content summaries of web databases by running a relative inexpensive sampling algorithm.

3.2 Incorporating Content Summary Extraction into SDARTS

We now explain low-level details of how we implemented the content summary generation algorithm of Section 3.1 into SDARTS.

The tool that implements the content summary generation is a J2EE-compliant servlet and can be easily installed in any web server that supports servlets. The servlet has to be initially configured with a classification scheme and its associated classifiers. These classifiers are generated automatically from a set of training documents, and they only have to be produced once. We include a comprehensive classification scheme with the associated classifiers as part of the toolkit. Of course, it is possible to create different classification schemes that fit the needs of each specific application. After this initial setup, this tool can be used with any web-accessible collection to generate content summaries.

To generate the content summary for a web-accessible collection, we have to first configure the SDARTS web wrapper for the collection, as described in Section 2.2. After this, the collection is SDARTS-compliant, so we can issue queries and retrieve results within the SDARTS protocol. Then, to generate an approximate content summary for the collection, we can use a simple web interface to start the probing process. This process returns the content summary of the collection in question after the construction of the approximate content summary has been completed.

As a result of this new content-summary extraction tool for remote web-accessible collections, the SDARTS web wrappers can now export high-quality fine-grained content summaries as well, just as the wrappers for locally available collections of plain-text and XML documents do. All SDARTS wrappers, regardless of whether the underlying collections are local or remote, are now able to provide metasearchers with rich metadata for sophisticated database selection.

4. SUPPORTING OAI COLLECTIONS

An emerging protocol, the *Open Archives Initiative* ("OAI" for short), defines an alternative interface and metadata that each collection should support to facilitate interoperability and metadata harvesting. Defining an interface between SDARTS and OAI is an attractive way to leverage both efforts for enhanced interoperability among a potentially large number of collections. In this section, we first briefly review the OAI protocol (Section 4.1). Then, we show how every SDARTS-compliant collection can now be made OAI-compliant automatically by exporting its metadata according to OAI (Section 4.2). Finally, we describe how we can wrap OAI collections to make them also SDARTS-compliant with minimal effort. In doing so, we also add search capabilities to OAI collections via SDARTS (Section 4.3).

4.1 Open Archives Initiative

The Open Archives Initiative (OAI) is a "low-barrier interoperability effort" [12], designed as an easy-to-implement and easy-to-use solution for metadata harvesting. The OAI differentiates mainly between two classes of participants:

- *Data Providers*, who use the OAI framework to export metadata about their collections.
- *Service Providers*, who use the metadata exported by the data providers to create value-added services on top of this data.

The only requirement for a data provider is to publish metadata about the collection using the Dublin Core Metadata Element Set [21]. All OAI implementers should support the Dublin Core and export the metadata records as XML documents. The OAI framework also allows, optionally, different metadata sets to be hosted under the same protocol. Each metadata set has an associated XML schema to validate its entries.

The OAI framework gives two main options for harvesting metadata from data providers. One option is to restrict the harvesting by date (e.g., "retrieve all the records that have been updated in the last three days"). This allows service providers to harvest metadata records incrementally and periodically update their repositories. The other option is to use the use the concept of *set*, defined as a group of items in a repository that are closely related. Hence, the set concept can be used to organize the metadata in a hierarchical scheme, and to separate the metadata of logically different collections hosted in the same repository (e.g., medical research vs. consumer health articles).

4.2 Exporting SDARTS Metadata using OAI

SDARTS includes tools to make locally-available text and XML collections searchable and to export metadata about them with minimal effort. Hence, a data provider can use SDARTS to add a SDARTS search and metadata interface to a collection with no programming whatsoever. To improve interoperability we decided to have the SDARTS-compliant collections export metadata using the OAI protocol as well. Hence, it is now possible for an information provider to use SDARTS as an off-the-shelf tool for publishing a text collection under SDARTS while exporting metadata using the OAI protocol at the same time.

To make SDARTS collections OAI-compliant, we map the SDARTS "Basic-1" attributes for documents [8][3] to Dublin Core elements, so that our OAI implementation will conform with the current OAI standard. In Table 2 we list the mapping of the different metadata elements. Since Dublin Core elements cover all the important metadata fields of SDARTS, we can publish the existing metadata using the Dublin Core elements, which are the core of the OAI metadata set. One drawback of our current implementation is the lack of support for all the Dublin Core elements in the Basic-1 attribute set (e.g., for the *Rights* element). However, since SDARTS inherits attribute-set extensibility from STARTS [8], in the future we could easily define mechanisms to allow for attribute-mapping extensibility as well as part of SDARTS.

Rather than just publish OAI metadata about the records (i.e., documents) stored in a SDARTS-compliant collection to make it OAI-compliant, we also generate and export the SDARTS content summary of the collection under OAI. We believe that this metadata can help the distributed search task over OAI collections as well. To export the SDARTS-specific metadata under OAI, we defined an XML schema that describes the structure of the SDARTS metadata. By doing this it is possible to export the SDARTS metadata directly under the OAI "umbrella" without modification. Since we have a different content summary for each subcollection accessible through SDARTS, we decided to consider each subcollection of SDARTS as an OAI set. Hence it is possible to export a "pointer" (i.e., a URL) to the content summary for each subcollection using the metadata "slots" that OAI provides to characterize each set.

In terms of implementation, the OAI server included in the SDARTS toolkit is based on the OCLC OAI Server [16]. In order to publish the URLs for the content summaries and remain completely OAI compliant, we had to make small modifications in the code relating to the OAI "Identify" verb. To assure platform independence, we used JDBC to interact with the underlying database that keeps the "OAI" data. This ensures that any relational database can be used to support the OAI server. In our implementation we use the MySQL [15] database management system, an open source project that is freely available for use.

[3]SDARTS inherits all the metadata defined by STARTS. Hence, what we refer to in this paper for simplicity as SDARTS metadata and content summaries are essentially *STARTS* metadata and content summaries.

Dublin Core	*SDARTS*
Title	Title
Creator	Author
Subject	Body-of-text
Description	Body-of-text
Format	Linkage-type
Identifier	Linkage
Source	No similar concept in SDARTS.
Type	No similar concept in SDARTS.
Language	Language
Publisher	No similar concept in SDARTS.
Contributor	No similar concept in SDARTS.
Date	This date could be a century or multiple years. No similar concept in SDARTS.
Relation	If recommended best practices are followed, this would map to Cross-reference-linkage. Not mapped in this implementation.
Coverage	Refers to a spatial or temporal period. No similar concept in SDARTS.
Rights	SDARTS does not have this concept on a document level.

Table 2: Mapping SDARTS's Basic-1 and OAI's Dublin Core attributes.

For ease of use, the OAI server that comes with SDARTS is distributed as a simple Web Application aRchive (WAR) file. The potential user has just to install this file on a Java servlet enabled web server and specify the location of the SDARTS database in the properties file. After this simple step, the OAI server is fully functional. To ensure minimum space utilization, the metadata is materialized in the database only when requested by an OAI "harvest" request.

Through the web-accessible server, it is directly possible to update the data available through the OAI server, and it is also possible to specify which SDARTS collections should export their metadata under the OAI server. These tasks are easily performed using a simple web interface.

4.3 Wrapping OAI Collections

We now describe how we can enhance collection interoperability by making each OAI collection SDARTS compliant. As we will see, the SDARTS wrapper for OAI collections is extremely simple to configure, since it exploits the uniform interface defined by OAI.

The OAI operations do not directly allow searching on different attributes, since the focus of the protocol is to facilitate metadata harvesting and not searching over them. This operation is left as "value-added" service provided by service providers, in contrast to the data providers who provide the actual document collections. The only fields that can be restricted during harvesting is the "date" field, to allow incremental harvesting of metadata, and the "set" field.

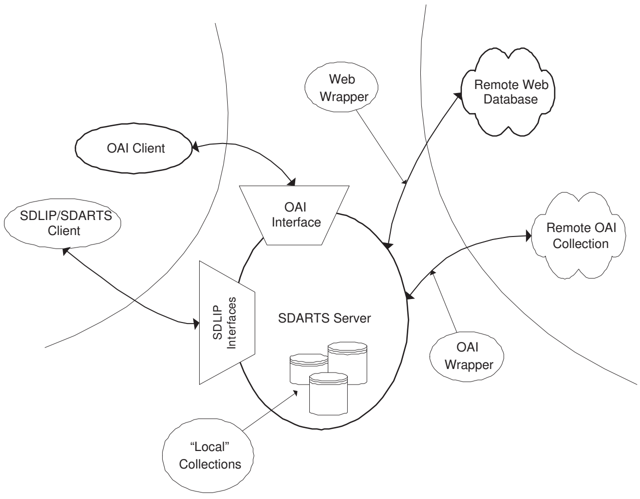

Figure 4: The SDARTS architecture: Interacting with multiple collection types and exporting metadata about them using different protocols.

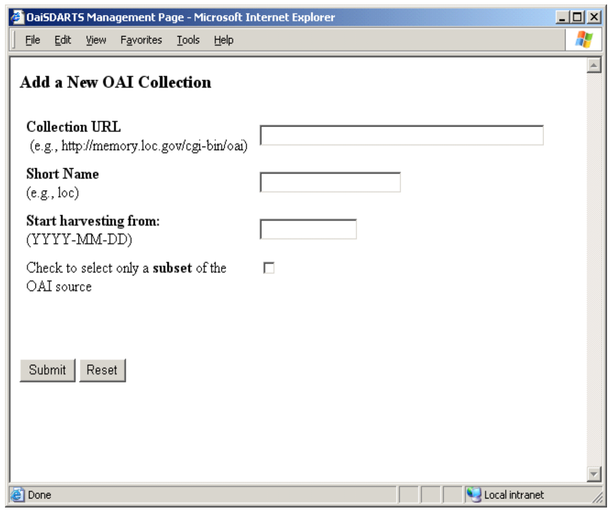

Figure 5: The user interface for "wrapping" an OAI collection and make it SDARTS compliant.

Hence, it is not possible to directly ask from an OAI collection to return the documents that have a specific keyword in the "title" field, for example.

For the metasearching task, however, the ability to perform search operations on different attributes is important. For example, a metasearcher might need to send a query to retrieve documents with specific keywords in the title, or to retrieve records that correspond to a given author. As we described above, such operations are not directly supported by the current version of OAI.

To overcome this restriction, we harvest the metadata from the remote OAI archives and then index and search them locally. The SDARTS XML wrapper (Section 2.2) provides the ability to index and search any local XML collection. The metadata records harvested from OAI collections are essentially XML documents with a uniform structure. Hence, we can fetch the OAI metadata records and store them locally as a collection of XML documents. After this, a SDARTS XML wrapper can be easily configured to make this XML collection fully SDARTS-compliant. This configuration just requires specifying an XSL file with the location in the XML files of the attributes of interest. Since all the OAI records have the same XML format, this configuration file had to be written only once and is now part of the OAI wrapper in SDARTS.

The only parameter that the OAI wrapper needs is the address of the remote OAI collection and which set (if there is a desire to specify one) it should retrieve. After giving these parameters through a simple web-accessible interface (Figure 5), the SDARTS administrator has to define how many records should be harvested and when. Then the process starts in the background, fetches XML OAI records from the remote OAI collection, stores them locally and then indexes them as an ordinary XML document collection. After retrieving the first documents and creating a local index of the OAI collection, the SDARTS administrator can update periodically the archive by sending the "update" command to the SDARTS server. This command retrieves the records that have been updated since the last harvesting, and the local index is updated accordingly.

One advantage of having a local copy of the metadata records is the ability to index and search virtually any attribute in the metadata records. This enables us to locate records that satisfy more complex queries, like "author contains Smith AND title contains bridge." Hence, any OAI collection can become transparently part of our distributed search testbed. Since we offer the ability to search the collections in our testbed, one can use SDARTS to query in more advanced ways existing OAI collections. Also, it is directly possible for an interested party to download the SDARTS toolkit and interact easily with the OAI collections of interest, without any additional effort.

Finally, one significant advantage of having OAI metadata as a local SDARTS-compliant collection is the ability to construct detailed content summaries of the OAI collections. Although these content summaries are constructed based only on the metadata and not based on the actual contents of the documents, these summaries can be of use for database selection. For example, in Table 3 we show a fraction of the content summary for the OAI-compliant "Virginia Tech Electronic Thesis and Dissertation" collection. We can see that the words "thesis" and "study" have much higher frequencies than other words, like "cancer," that do not correlate well with the contents of this collection. By comparing this content summary with the one that we extracted from PubMed (Table 1), we can see that the word distribution can be used to distinguish between the two collections, which host documents of completely different type. For example, the word "cancer" in PubMed has high frequency, while the frequency of the same word in the thesis repository is really low since these theses do not focus on medical issues. Hence, a metasearcher that has to pick one

Word	*Document Frequency*
thesis	2,945
study	1,823
surface	275
operation	174
...	...
basketball	2
cancer	1
bone	1

Table 3: A small fraction of the content summary for the OAI-compliant "Virginia Tech Electronic Thesis and Dissertation" collection.

of these two collections for a query on "cancer" can examine these content summaries and direct the query to the PubMed database. Thus, the additional metadata and in particular the SDARTS content summaries of the existing OAI collections can be beneficial for distributed search.

5. CONCLUSION

We have described in this paper how we extended SDARTS, a protocol and toolkit designed to facilitate metasearching. Until now, the SDARTS toolkit had the capability to index, search, and export metadata for any text or XML collection of documents that was available locally. Also, it was possible to wrap "external" web collections and search them using a uniform interface. However, no content summaries for web collections were available, since these collections rarely export any metadata whatsoever. Hence, in this paper we introduced a new tool into the SDARTS toolkit that automates the task of extracting content summaries from web collections. By extracting these content summaries we believe that we now make metasearching over web collections an easier task.

Additionally, we decided to make any SDARTS-compliant collection also OAI-compliant by exporting metadata under the OAI protocol. Using OAI, we export not only the required Dublin Core metadata (with some exceptions), but also the SDARTS content summaries for the collections. This makes it possible for potential users to use these content summaries directly just by consulting the SDARTS metadata interface (using either SDLIP or OAI).

We also described in the paper how we created a universal OAI wrapper to make OAI-compliant collections also SDARTS-compliant, adding search capabilities on top of them. Additionally, as an interesting by-product of our approach, SDARTS generates rich content summaries for the SDARTS-wrapped OAI collections. These summaries then allow metasearchers to do sophisticated database selection over OAI collections as well, just as over any other SDARTS-compliant collections.

The SDARTS toolkit, related documentation and code are publicly available from `http://sdarts.cs.columbia.edu`.

6. REFERENCES

[1] International Standard Maintenance Agency. *Z39.50 Maintenance Agency Page*. ISMA, 2000. Available at `http://lcweb.loc.gov/z3950/agency/`.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. Available at `http://www.ics.uci.edu/~mlearn/-MLRepository.html`.

[3] James P. Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 479–490, 1999.

[4] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[5] William W. Cohen. Learning trees and rules with set-valued features. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96) Eighth Conference on Innovative Applications of Artificial Intelligence (IAAI-96)*, pages 709–716, 1996.

[6] Nick Craswell, Peter Bailey, and David Hawking. Server selection on the World Wide Web. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL 2000)*, pages 37–46, 2000.

[7] Daniel Dreilinger and Adele E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222, 1997.

[8] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. *STARTS*: Stanford proposal for Internet meta-searching. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, pages 207–218, 1997.

[9] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. *GlOSS*: Text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, June 1999.

[10] Noah Green, Panagiotis G. Ipeirotis, and Luis Gravano. SDLIP + STARTS = SDARTS: A protocol and toolkit for metasearching. In *Proceedings of the the First ACM+IEEE Joint Conference on Digital Libraries (JCDL 2001)*, pages 207–214, 2001.

[11] Panagiotis G. Ipeirotis and Luis Gravano. Summarizing and searching hidden-web databases hierarchically using focused probes. Technical Report CUCS-015-01, Columbia University, Computer Science Department, 2001.

[12] Carl Lagoze and Herbert Van de Sompel. The Open Archives Initiative: Building a low-barrier interoperability framework. In *Proceedings of the the First ACM+IEEE Joint Conference on Digital Libraries (JCDL 2001)*, pages 54–62, 2001.

[13] Kathleen McKeown, Shih-Fu Chang, James J. Cimino, Steven Feiner, Carol Friedman, Luis Gravano, Vasileios Hatzivassiloglou, Steven Johnson, Desmond A. Jordan, Judith Klavans, André Kushniruk, Vimla L. Patel, and Simone Teufel. PERSIVAL, a system for personalized search and

summarization over multimedia healthcare information. In *Proceedings of the the First ACM+IEEE Joint Conference on Digital Libraries (JCDL 2001)*, pages 331–340, 2001.

[14] Weiyi Meng, King-Lup Liu, Clement T. Yu, Xiaodong Wang, Yuhsi Chang, and Naphtali Rishe. Determining text databases to search in the Internet. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB'98)*, pages 14–25, 1998.

[15] `http://www.mysql.com/`, 2001.

[16] ALCME: OAICat project, `http://alcme.oclc.org/oaicat/index.html`, 2000.

[17] The Open Archives Initiative, `http://www.openarchives.org/`, 2000.

[18] Andreas Paepcke, Robert Brandriff, Greg Janee, et al. Search middleware and the Simple Digital Library Interoperability Protocol. *D-Lib Magazine*, 6(3), 2000.

[19] Erik Selberg and Oren Etzioni. Multi-Service search and comparison using the MetaCrawler. In *Proceedings of the Fourth International World Wide Web Conference (WWW4)*, 1995.

[20] Atsushi Sugiura and Oren Etzioni. Query routing for web search engines: Architecture and experiments. In *Proceedings of the Ninth International World Wide Web Conference (WWW9)*, 2000.

[21] Stuart Weibel, Jean Godby, Eric Miller, and Ron Daniel Jr. OCLC/NCSA metadata workshop report, March 1995. Accessible at `http://-www.oclc.org:5047/oclc/research/publications/-weibel/metadata/dublin_core_report.html`.

[22] Jinxi Xu and James P. Callan. Effective retrieval with distributed collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'98*, pages 112–120, 1998.