Semi-automatic Wrapper Generation for Internet Information Sources *

Naveen Ashish and Craig Knoblock
Information Sciences Institute and
Department of Computer Science
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{ashish,knoblock}@isi.edu
http://www.isi.edu/sims/{naveen,knoblock}

Abstract

To simplify the task of obtaining information from the vast number of information sources that are available on the World Wide Web (WWW), we are building tools to build information mediators for extracting and integrating data from multiple Web sources. In a mediator based approach, wrappers are built around individual information sources, that provide translation between the mediator query language and the individual source. We present an approach for semi-automatically generating wrappers for structured internet sources. The key idea is to exploit formatting information in Web pages from the source to hypothesize the underlying structure of a page. From this structure the system generates a wrapper that facilitates querying of a source and possibly integrating it with other sources. We demonstrate the ease with which we are able to build wrappers for a number of Web sources using our implemented wrapper generation toolkit.

1. Introduction

We are building information agents or mediators to gather and integrate information from multiple World Wide Web sources. The mediator [18, 3] approach has been used to integrate information from distributed heterogeneous database systems, where a mediator insulates the user from problems caused by different locations, query languages and protocols of the different databases. We are extending the mediator approach to integrate information from multiple Web sources. Our approach is to take several related Web sources in a particular *domain* of interest (e.g., finance, government, or real-estate) and provide integrated access to multiple Web sources through a mediator.

For example, we can use a mediator to provide integrated access to multiple Web sources that provide information on countries in the world. An excellent Web source is the *CIA World Fact Book* [1], which provides information on the geography, economy, government, etc., of every country. Other interesting sources include the Yahoo listing of countries by region from where we can obtain information such as what countries are in Europe, the Pacific Rim, etc. Another interesting source is the on-line listing of country corruption rankings. A user could query a mediator that provides access to the above sources to answer queries such as ``Find the *Economic Overview*, *Telephone System* and *Corruption Rankings* of all countries in the Pacific Rim''. The mediator would determine what sources can be used to answer the query, retrieve information from these sources, and present the integrated result to the user. There are several other research projects that are working on integrating Web-based sources, The most prominent ones include InfoSleuth [4], the OBSERVER project [14], the Information Manifold [10] and the Internet Softbot [6].

An essential component in a mediator architecture is a *wrapper* around each individual data source (see Figure 1), which accepts queries from the mediator, translates the query into the appropriate query for the individual source,

*This work is supported in part by the University of Southern California Integrated Media Systems Center (IMSC) - a National Science Foundation Engineering Research Center, by the Rome Laboratory of the Air Force Systems Command and the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-94-C-0210, by the National Science Foundation under grant number IRI-9313993 and by the DARPA Fort Huachuca Contract DABT63-96-C-0066. The views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, RL, NSF or any person or agency connected with them.

[1] http://www.odci.gov/cia/publications/nsolo/wfb-all.htm

performs any additional processing if necessary, and returns the results to the mediator. To make the integration of Web sources using the mediator approach feasible, wrappers are needed for all of the Web sources to be accessed. Wrappers for Web sources would accept a query from the mediator, fetch the relevant pages from that source, extract the requested information from the retrieved pages and return the results to the mediator. Essentially the wrappers make the Web sources look like databases that can be queried through the mediator's query language. The basic techniques applied in database integration using mediators can then be applied to Web sources integration. It is however impractical to construct wrappers for Web sources by hand for a number of reasons:

- The number of information sources of interest is very large, even within a particular domain.
- Newer sources of interest are added quite frequently on the Web.
- The format of existing sources often changes.

We report on the development of an implemented wrapper generation toolkit that provides a semi-automatic, interactive wrapper generation facility for Web sources. It should be noted that building wrappers is just one of the challenges in building the kinds of information mediators for the Web, that we envision. Problems lie in several other areas such as modeling the information sources, resolving semantic heterogeneity amongst different sources, query planning to gather the requested information from different sites, and intelligently caching retrieved data, to name a few. The focus of this paper is solely on wrapper generation.

The rest of this paper is organized as follows. Section 2 provides an overview of the different kinds of information sources on the Web. Section 3 describes how we semi-automatically generate wrappers. Section 4 presents experimental results to demonstrate the effectiveness of our techniques for wrapper generation. Section 5 describes related work. Section 6 presents future directions and conclusions.

2. Types of Web Information Sources

We categorize the types of pages from Web sources into three classes: multiple-instance sources, single-instance sources, and loosely-structured sources. Certain sources provide information in multiple pages, all conforming to the same format. We call such sources *multiple instance* sources. Consider a source such as the *CIA World Fact Book*. This source provides information on each of the 269 countries in the world, with information for each country presented on a separate page for that country. The information on each page is presented in a semi-structured manner since each page can be clearly sub-divided into distinct sections with headings labeling the beginning of each section. Also, the information on all pages is presented in *exactly* the same format. A page for one country is shown in Figure 2. There are clearly identifiable sections such as Geography, Area, Land boundaries etc., on each page. For each individual page we would like the wrapper to able to handle queries about one or more sections in the page. For example ``Find the *Land boundaries* and *Area* of France''. This wrapper will in turn allow a mediator to handle aggregate queries (spanning multiple countries) such as ``Find the *National Product*, and *Defense Expenditures* of all countries in Europe.''

There are a number of sources on the Web that fall in the multiple instance category, such as the National Science Foundation (NSF) Grants database [2], the General Services Administration (GSA) On-line Shopping database [3], the NSF Funding Opportunities database, Genetics databases such as OMIM [4], or the Air Force Fact Sheets [5] to name a few. It might be argued that for this category of sources, the information that is put on-line often comes from a database itself. Thus we should query the databases directly. Unfortunately for most of these sources, access to the underlying databases is simply not permitted or might be allowed only with a license fee to query the database. However the information put on-line is readily and freely accessible, which makes a case for building wrappers in order to query these sources.

Another category is that of semi-structured *single instance pages*. There are numerous sources on the Web that contain useful information in a semi-structured form, but on a single page. To name a few, consider the CoopIS 96 proceedings page, list of AAAI Fellows or the Yahoo list of countries by region. Consider the CoopIS 96 proceedings page [6]. The page is organized into clearly identifiable sections, with a heading for each section (such as Classification and Ontologies, or Data Integration etc.). Each section starts with the Chair of that section followed by papers presented in that section. From such a page we would like a wrapper to be able to answer queries such as ``Find the names of all people who chaired a session in CoopIS 96'' and expect the wrapper to extract and return the list of chairs i.e., "Witold Litwin, James Geller, Klemens Bohm ...".

Finally there are pages that are more loosely structured, such as a personal homepage. For such cases, i.e., in the absence of clearly identifiable sections with headings, the ex-

[2] http://cos.gdb.org/best/fedfund/nsf-intro.html
[3] http://www.fss.gsa.gov/cgi-bin/advantage!?38
[4] http://www3.ncbi.nlm.nih.gov/Omim
[5] http://www.af.mil/pa/indexpages/fs_index.html
[6] http://sunsite.informatik.rwth-aachen.de/dblp/db/conf/coopis/coopis96.html#ScheuermannLC96

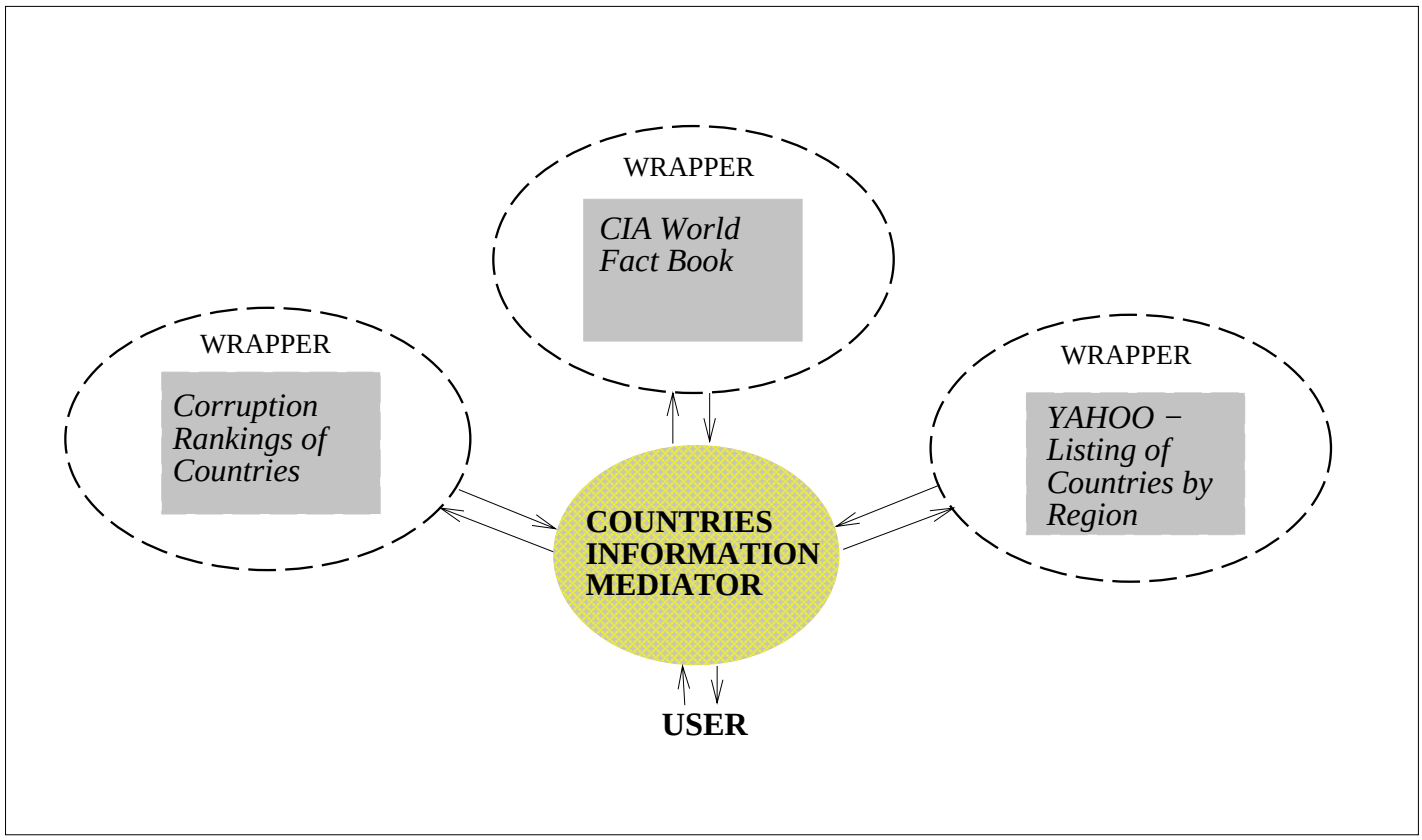

Figure 1. Role of wrappers in providing integrated access to multiple information sources

France

Geography

Location: Western Europe, bordering the Bay of Biscay and English Channel, between Belgium and Spain southeast of the UK; bordering the Mediterranean Sea, between Italy and Spain

Map references: Europe

Area:
total area: 547,030 sq km
land area: 545,630 sq km
comparative area: slightly more than twice the size of Colorado
note: includes Corsica and the rest of metropolitan France, but excludes the overseas administrative divisions

Land boundaries: total 2,892.4 km, Andorra 60 km, Belgium 620 km, Germany 451 km, Italy 488 km, Luxembourg 73 km, Monaco 4.4 km, Spain 623 km, Switzerland 573 km

Coastline: 3,427 km (mainland 2,783 km, Corsica 644 km)

Figure 2. Snapshot of a page from the *CIA World Fact Book*

traction task becomes much harder. Also the use of fancy graphics or images for information presentation makes the task of building a wrapper for that source more difficult.

In this paper we focus on semi-automatically building wrappers for semi-structured sources, in both the multiple-instance and single-instance categories. For loosely structured sources or sources with complicated graphics we have to build wrappers manually. However it is the large number of sources in the semi-structured category, and the wealth of information that can be obtained from them that has motivated us to automate the task of wrapper generation for such sources.

3. Approach for Automated Wrapper Generation

This section describes our approach for generating wrappers for Web sources. We have attempted to automate the process of building wrappers as much as possible. The following steps are involved in generating a wrapper for a new Web source:

- *Structuring the source*. This involves identifying sections and sub-sections of interest on a page.
- *Building a parser for the source pages*. After structuring the source we build a parser that can extract selected sections from a page from the source.
- *Adding communication capabilities between the wrapper, mediator and web sources*. The mediator that integrates several sources must be able to communicate with the wrappers for these sources. Also, the wrappers must communicate with Web sources to retrieve data in order to answer queries.

We describe these steps in detail below.

3.1. Structuring the Source

In specifying the structure of a page on the Web, two things need to be clearly identified.

1. *Tokens of interest on a page*. By tokens we mean words or phrases that indicate the heading of a section, such as `Geography`, `Economy`, or `Total Area` on the *CIA World Fact Book* page. A heading indicates the beginning of a new section; thus identifying headings identifies the sections on a page.
2. *The nesting hierarchy within sections*. Once a page has been decomposed into various sections, we have to identify the nesting structure of the sections. For instance a *CIA World Fact Book* page comprises of the sections Geography, People, Economy, Government and Transportation. The Geography section in turn is broken down into the sections Area, Land boundaries, etc., while Area contains land area, total area, etc.

The structuring task can be done automatically or with minimal user interaction. The key idea here is that a program analyses the HTML and other formatting information in a sample page from the source and guesses the interesting tokens on that page. The system also uses the formatting information to guess the nesting structure of the page. The heuristics used for identifying important tokens on a page and the algorithm used to organize sections into a nested hierarchy are an important contribution of this work. We describe them in more detail below.

3.1.1 Identifying Tokens

Tokens identifying the beginning of a section are often presented in bold font in HTML. They may also be written entirely in upper case words, or may end with a colon. We can generate a lexical analyzer that searches a page for such tokens using LEX [13], a lexical analyzer generator. In Table 1 we list the regular expressions given as specifications to LEX to identify tokens indicating headings on a page. From these specifications we generate a lexical analyzer that identifies words or phrases conforming to the regular expressions. When structuring any page, the system is able to identify headings that are formatted in any of the ways listed in Table 1. For instance, given a page from the *CIA World Fact Book* the system is able to identify the tokens of interest such as `Geography`, `Land boundaries`, `Area` etc. Since each token marks the beginning of a section on a page, at the end of the above tokenizing step all the different sections on a page have been identified.

3.1.2 Determining the Hierarchical Structure

The next step is to obtain the nesting hierarchy of sections on the page, i.e., what sections comprise the page at the top level, what sub-sections comprise other sections in the page, etc. As with the tokenizing step, the nesting hierarchy can be obtained in a semi-automatic fashion for a large number of pages. We have developed an algorithm that, given a page with all sections and headings correctly identified, outputs a hierarchy of sections. The following two simple heuristics are used:

1. Font Size -The font of the heading of a sub-section is generally smaller than that of its parent section.
2. Indentation - Indentation spaces (which can be detected from raw text or HTML tags) are often used to indicate that one section is a subsection of another.

Regular Expression given as LEX Specification	*Description*	*Example Heading*
``<''[bB][^<>]*``>''[^\n]+``<''/[bB]``>''	Headings in bold tags	<a href=```>Chair
``<''[hH][0-6][^<>]*``>''[^\n]+``<''/[hH][0-6]``>''	Headings with font size	Geography
``<STRONG>''[^\n]+``</STRONG>''	Headings in Strong Tags	<STRONG>Area</STRONG>
``<Strong>''[^\n]+ ``</Strong>''	Strong tags in different case	<Strong>Population</Strong>
``<strong>''[^\n]+``''	Strong tags in lower case	Deadlines
[A-Za-z0-9\-_]+[:]	Words ending in colon	IRS NUMBER:
``<''[iI][^<>]*``>''[^\n]+``</''[iI]``>''	Italicized words	total area:

Table 1. Heuristics for identifying tokens when structuring a page

```
current_node= make_new_tree(); /* returns a  node that is the root of a new tree */
while(more_headings){

    new_node=construct_node(heading); /* makes a new node for the new section */
    while ((size_of(current_node) <= size_of(new_node)) or
       (indentation_of(current_node) >= indentation_of(new_node))){
                    /* search for the immediate parent section  of the new section */
           current_node=parent_of(current_node);
        }
    make_rightmost_child(current_node,new_node);
                    /* make the new section the rightmost child of its immediate parent */
    current_node=new_node;
}

generate_grammar();
/ * a procedure that from the tree constructed above, for each node N with ordered children
C1, C2 ... , Cm outputs a grammar rule of the form N --> C1 C2 ..... Cm */
```

Figure 3. Algorithm to obtain nesting hierarchy

Using the procedure shown in Figure 3, the system outputs a grammar describing the nesting hierarchy of sections in a page. This procedure first builds a tree that reflects the nesting hierarchy of sections. We construct a node for each heading that identifies a new section, and make this node a child of the section that should be its immediate parent based on the font size and indentation of the section headings. The children of each node are ordered, i.e., they appear in the same order in which the corresponding sections appear on the page. When all nodes for all sections have been placed in the tree, the procedure outputs grammar rules for each node in the tree, essentially stating that the section at each node has as sub-sections all its immediate children in the tree (and in the order in which they appear in the tree). For instance, for pages from the *CIA World Fact Book* the grammar output is shown in Figure 4.

It is possible for the system to make mistakes when trying to identify the structure of a new page. Based on the heuristics listed in Table 1, the system can highlight tokens erroneously (that is identify some words or phrases as tokens when they are not, or fail to identify phrases that are headings or tokens, but do not conform to any of the regular expressions in Table 1). We have provided a facility for the user to interactively correct the system's guesses. Through a graphical interface the user can highlight tokens that the system misses, or delete tokens that the system erroneously chooses. The user can similarly correct errors in the system-generated grammar that describes the structure of the page.

3.2. Building a Parser for the Source Pages

The next step is to generate a parser for pages from the source. Given a page from the source, such a parser can extract any selected section(s) from the page. For instance a parser for pages from the *CIA World Fact Book* can extract sections such as Geography.Area (the "." indicates that Area is a subsection of Geography in the spirit of complex objects) i.e., the Area sub-section within the Geography section from the page for any country.

```
<h3>Geography<h3>   {return(GEO_HEADING);}
<b>Location:</b>   {return(LOC_HEADING);}
<b>Map references:</b>   {return(MAP_HEADING);}
<b>Area:</b> {return(AREA_HEADING);}
<i>total area:</i>   {return(TOT_HEADING);}
...

. {return(TEXT);}
\n   {return{TEXT);}
```

Figure 5. LEX Specifications for CIA Page

Such a parser can be automatically generated, since all of the grammatical and lexical information needed to parse the page is obtained at the structuring step. The compiler generator YACC [9] and the tool LEX are used for this purpose. The tokens identified in the structuring step are directly input as specifications to LEX to generate a lexical analyzer for a page from the source. For instance the tokens identified in the *CIA World Fact Book* page are Geography, Location:, Map references:, Area:, total area: etc., and the specifications given to LEX to generate a lexical analyzer for a page from the *CIA World Fact Book* are shown in Figure 5.

The tool YACC can generate a parser for a language given grammar rules that specify valid sentences in the language. We directly translate the grammar rules describing the overall structure of the page, obtained at the end of the structuring step, into a YACC specification. The parser generated can parse valid "sentences" i.e., pages from the source. Figure 6 shows what the rules specified to YACC to parse pages from the *CIA World Fact Book* look like. For instance the first part of the first rule states that a single page is comprised of the Geography section, People section etc. The second part of the rule shows YACC code for storing and manipulating parsed data. With these specifications we use LEX and YACC to generate a parser for pages from the source.

3.3. Adding Communication Capabilities between the Wrapper, Mediator and Web Sources

Given a query, a wrapper for a Web source should be able to fetch the pages containing the requested information from the Web source. Also some mechansim is needed for communication between the mediator and the wrapper as they are separate processes, possibly running at different locations. The following communication functionality thus needs to be added to the wrapper.

1. *Identifying network locations of page(s) needed to answer a query*. For sources with just a single page this is straightforward i.e., the URL for that page is known to the wrapper. For sources with multiple pages, a mapping between a query and the URL of the relevant page might be required. For instance for the *CIA World Fact Book* there is a one to one mapping between the country name and the URL of the page for that country. This mapping can be obtained from the index page for the CIA source. For the GSA database the *part number* appears at the end of the URL for that source to point to the page for that part.

   To provide the capability of determining the network location of the page relevant to a query, the user specifies a *mapping function* which takes necessary arguments from a query (eg. *country name* from a query on

```
CIApage –> Geography People Government Economy Transportation

Geography –> Location Map_references Area Land_boundaries Coastline ..

Area –> total_area land_area comparative_area
...
```

Figure 4. Nesting Hierarchy for CIA Page

```
CIApage          :Geographysection Peoplesection Governmentsection Economysection Transportsection
                 {strcpy($$,$1); strcat($$,$2); strcat($$, $3); strcat($$,$4); }

Geographysection : Locationsection Maprefsection Areasection Landboundariessection...
                 {strcpy($$, $1); ... }

Areasection      :totalareasection landareasection compareasection
                 {strcpy($$,$1); strcat($$,$2); strcat($$,$3); }
Locationsection  : Locationheading Text
                 {strcpy($$, $1); strcat($$,$2); }

Maprefsection    : Maprefheading Text
                 {strcpy($$, $1); strcat($$,$2); }
Locationheading  : LOC_HEADING
                 { strcpy($$, yytext); }

Maprefheading    : MAP_HEADING
                 { strcpy($$, yytext); }

...
```

Figure 6. YACC specifications for CIA page

the CIA source) and constructs a URL pointing to the page to be fetched.

2. *Capability to retrieve data over the network*. Currently we are using PERL scripts for the purpose of making HTTP connections to the Web information sources and retrieving data from them.

3. *Communication between the mediator and wrapper*. We are using the agent communication language KQML [7] for the purpose of providing interprocess communication between the mediator and a wrapper.

Adding the above functionality is the final step in generating a wrapper for a new source. The parser for pages from a Web source plus the above communication functionality results in a complete wrapper for that Web source.

4. Results

We have applied the wrapper generator to the task of generating wrappers for a variety of internet sources. We present experimental results to provide an idea of the effort required to generate a wrapper for a new source. The step that is most difficult to automate when generating a wrapper is the first step where we obtain the structure of a page or sample pages from the source. Generating the parser is then done automatically and defining a mapping function from queries to URLs of relevant for sources with multiple pages requires comparatively little effort on part of the user. It is thus the structuring step that dominates the time and effort needed to build a wrapper for a new source.

We used the wrapper generator to build wrappers for several internet sources and to evaluate the effectiveness of the heuristics we use for structuring a new page automatically. To provide a quantitative measure of the effectiveness of the heuristics, we define what we call *correction steps*. During structuring a page, each time the user has to manually correct a token (i.e., add or delete a token) or correct a rule in the grammar describing the nesting hierarchy of sections, it is counted as one correction step. The total number of correction steps made before the page is completely structured provides an estimate of how hard it is to automatically structure that page. We also provide the time taken to generate the wrapper for each source. This would of course vary from

Multiple instance sources	*Correction steps*	*Time in min*	*Single instance sources*	*Correction steps*	*Time in min*
1. The CIA World Fact Book.	0	2	1. CoopIS 96 Proceedings page	1	3
2. GSA On-line Shopping database.	2	5	2. AAAI-97 conference homepage	3	3
3. The NSF database.	0	5	3. List of US Universities by state	6	4
4. The OMIM Genetics database	4	4	4. List of AAAI Fellows by year	0	1
5. Hoover Company Profiles	2	4	5. Computer Science Job Listings	6	5
6. The Internet Movie Database	3	6	6. SIGMOD Record page	4	3
7. The Air Force Library Fact Sheets	0	2	7. US Air Force Organization page	0	1

Table 2. Experimental Results showing the effort and time to build wrappers for different sources

user to user. Nevertheless, the results give a sense for approximately how long it might take to generate wrappers using this toolkit.

Table 2 demonstrates the ease with which we built sources for a dozen internet sites, from both the multiple instance and single instance categories. We provide the number of correction steps to structure a sample page from each source as well as the total time (in minutes) taken to build a wrapper for that source. The results are extremely encouraging. Several sources require almost none or very few correction steps to structure them, thus showing that the heuristics for structuring pages are quite successful. Also, it takes only a few minutes to generate a wrapper for most new sources. Such a toolkit is thus extremely useful as it provides a convenient and quick way to generate wrappers for new sources of information on the Web and then integrate them via a mediator. We successfully integrated several sources in the countries information domain, such as the *CIA World Fact Book*, Yahoo listings of countries by region etc. using the Ariadne system, which is a descendant of the SIMS [3] information mediator that addresses the problem of integrating Web sources. We were then able to pose queries to Ariadne such as " Find the *External debt* and *Defense expenditures* of all countries in EEC." The answer given by the mediator is shown in Figure 7.

5. Related work

The work of [8] is on wrapper generation, although the techniques discussed are for specifying wrappers for various kinds of sources such as relational databases and legacy systems besides Web sources. A *template based* approach is used where a user provides actions that execute when a query matches a certain template or format. Wrapper generation for Web sources is also discussed in [12]. The focus of their work is very similar to ours i.e., building wrappers for Web sources to be integrated by a software agent. However, they follow a very different approach since they are working on building an Internet shopping agent and they focus on pages that contain items for sale. As a result they make much stronger assumptions about the type of information they are looking for and use that information to hypothesize the underlying structure. Their approach cannot generate wrappers for more general types of pages, such as the *CIA World Fact Book*.

Several researchers are also investigating the topic of querying semi-structured data, with a particular focus on data in Web pages. The work of [1] focuses on isolating the essential aspects of semi-structured data, and also surveys some proposals for models and query languages for such data. The work in [5] proposes a language for querying data that has a tree like structure and also discusses optimization issues for such a language. Lorel [2] is a query language for semi-structured data. It is based on the OEM data model and has been implemented on top of the O2 object-oriented database system. Another effort on using an object-oriented database to manage SGML data is described in [17]. W3QS [11] is a system for SQL like querying for the Web. The system interfaces to user programs and UNIX services for analyzing and filtering semi-structured information from Web servers. The work in [15] describes a query language WebSQL for querying Web sources by exploiting the structure and topology of the document networks. Most of the above efforts are concerned with issues such as the development of data models and query languages

Country	EXTERNAL_DEBT	DEFENSE_EXPENDITURES
AUSTRIA	$21.5 billion (1994 est.)	exchange rate conversion – about $1.8 billion, 0.9% of GDP (1994)
BELGIUM	$31.3 billion (1992 est.)	exchange rate conversion – $3.9 billion, 1.8% of GDP (1994)
DENMARK	$40.9 billion (1994 est.)	exchange rate conversion – $2.7 billion, 1.9% of GDP (1994)
FINLAND	$30 billion (December 1993)	exchange rate conversion – $1.86 billion, about 1.9% of GDP (1994)
FRANCE	$300 billion (1993 est.)	exchange rate conversion – $47.1 billion, 3.1% of GDP (1995)
GERMANY	$NA	exchange rate conversion – $40 billion, 1.8% of GNP (1995)
GREECE	$26.9 billion (1993)	exchange rate conversion – $4.1 billion, 5.4% of GDP (1994)
IRELAND	$20 billion (1994 est.)	exchange rate conversion – $500 million, 1.3% of GDP (1994)

Figure 7. Answer to query involving multiple Web sources

for semi-structured data, defining formal semantics for the proposed languages and implementation issues. However, the Web is the largest and ever growing source of semi-structured data. The focus of our work is on the generation of wrappers which can facilitate database-like querying of semi-structured data retrieved directly from Web servers as opposed to efforts that address the management of semi-structured data stored locally. The main contribution of our work is the development of heuristics by which the system can hypothesize the structure implicit from the formatting information in the source pages. Once the correct structure is obtained a wrapper for the source can be generated without much effort or time.

6. Future work and conclusions

We have presented the ideas and results of our approach for automatically generating wrappers for Web sources. We have clearly separated the tasks in building wrappers that are specific to a particular Web source such as structuring the source, and tasks which are repetitive for any source (and can thus be done by the system) such as generating a parser from the structure of a page and adding communication capabilities. The main contribution of our work is automating the structuring step, through the use of heuristics for determining the structure by exploiting formatting information in pages from the source. Our ideas appear to be effective for many semi-structured kinds of sources. However we need more advanced wrappers to be able to broaden the scope of sources we can generate wrappers for and also to be able to handle more finer grained queries. Currently we are working on enhancing the wrappers with the following capabilities:

- *Learning new tokens by examples:* It is possible that while structuring a page, the system is unable to identify tokens on the page if they do not conform to any of the regular expressions in Table 1. We are working an adding capabilities to the system to quickly *learn* the structure of a new kind of heading from a few user examples. We are applying techniques from inducing Hidden Markov models (HMMs), describing the tokens, from corpora of positive examples. The basic idea, described in [16] is to start with an HMM accepting only the initial tokens marked by the user. Then, states in the HMM are *merged* to yield a generalized model that can be used to identify the remaining tokens in the page. The system can then identify the remaining tokens in the page automatically.

- *Handling Tables:* A challenging problem is to automatically build parsers for information in tables. The hard problem here is to determine exactly what is contained in the different rows and columns of the table and then build a parser to extract information from it.

- *Handling finer grained queries:* Consider the Land boundaries section on a CIA World Fact Book page. Currently the wrapper cannot handle queries such as `''Find the names of all countries bordering France.''` as the parser does not have enough knowledge of the structure *within* the Land boundaries field to decompose that field into pairs of countries and corresponding border lengths. We are currently investigating using machine learning techniques where the user gives a few examples highlighting items of interest within a field and the system is eventually able to learn the structure within that field.

We are using the wrapper generator system to generate wrappers for semi-structured sources and are working on making the system more advanced and capable of handling more kinds of Web sources. Generation of wrappers is very useful in meeting our broader goal of integrating Web sources via a mediator, by which we hope to simplify the task of obtaining information from the already numerous and ever growing information sources on the Web.

7. Acknowledgements

We would like to thank Steve Minton and other members of the SIMS and Ariadne projects for their helpful contributions to this work. We also wish to thank Vipul Kashyap of the InfoSleuth project at MCC for suggestions on future enhancements.

References

[1] S. Abiteboul. Querying semi-structured data. In *ICDT (invited talk)*, 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Weiner. The Lorel query language for semistructured data. *Journal on Digital Libraries*, To appear.

[3] Y. Arens, C. A. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration*, 6(2/3):99–130, 1996.

[4] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Semantic integration of information in open and dynamic environments. Technical Report MCC-INSL-088-96, MCC, Austin, Texas, 1996.

[5] P. Buneman, S. Davidson, and G. H. D. Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, 1996.

[6] O. Etzioni and D. S. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7), 1994.

[7] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. AAAI/MIT Press, Menlo Park, CA, in press.

[8] J. Hammer, M. Brennig, H. Garcia-Molina, S. Nesterov, V. Vassalos, and R. Yerneni. Template-based wrappers in the tsimmis system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (Demonstration Track)*, Tucson, AZ, 1997.

[9] S. C. Johnson. Yacc: Yet another compiler compiler. Technical Report CSTR 32, AT&T Bell Laboratories, 1978.

[10] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Working Notes of the AAAI Spring Symposium on Information Gathering in Heterogeneous, Distributed Environments*, Technical Report SS-95-08, AAAI Press, Menlo Park, CA, 1995.

[11] D. Konopnicki and O. Shemueli. W3QS: A query system for the World Wide Web. In *Proceedings of the 21st International Conference on Very Large Databases*, Zurich, Switzerland, 1995.

[12] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, Japan, 1997.

[13] M. E. Lesk. Lex - a lexical analyzer generator. Technical Report CSTR 39, AT&T Bell Laboratories, Murray Hill, New Jersey, 1975.

[14] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer:an approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS '96)*, June, 1996.

[15] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the world wide web. In *Symposium on Parallel and Distributed Information Systems*, Miami, Florida, 1996.

[16] A. Stolcke and S. Omohundro. Inducing probabilistic grammars by bayesian model merging. In R.C.Carrasco and J. Oncina, editors, *Grammatical Inference and Applications*, pages 106–118. Springer, 1994.

[17] V.Christophides, S.Abiteboul, S.Cluet, and M.Scholl. From structured documents to novel query facilities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, 1994.

[18] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, March 1992.