Intelligent Wrapping of Information Sources in an Electronic Commerce Environment

Sebastian Pulkowski
Institute for Program Structures and Data Organization, University of Karlsruhe, Germany
E-mail: pulkowski@ira.uka.de

1 Introduction

The World Wide Web can be seen as one big virtual library. Information about documents or even the documents themselves in electronic format can be found on nearly every subject area. Thus literature search and delivery is a rapidly expanding market. Today almost all booksellers and publishers place their offers on the Internet, and intermediaries that catalogue and index documents for search assist users in the retrieval of relevant information. Almost all of these do so to make a profit and, consequently, charge users and/or providers for their services.

The problem facing a customer searching for information is to become knowledgeable about all these sources, find the most suitable ones and combine all sources into a search and delivery process that meets his or her needs.

The UniCats [2] project at the Universität Karlsruhe solves these problems by means of user agents and traders. These agents only submit the requests to selected information sources, aided by the users profiles. The selection is done by so-called traders, which hold technical and textual information about the sources.

The system mainly relies on wrappers to adapt the data sources such that user agents and traders can function as desired. The wrapper translates the user request to the source and the source results to the user. User agents and traders expect that a wrapper will carry out a request in its entirety, without further interaction on their part. If we assume that the sources are provided in the form of a set of statically or dynamically generated HTML pages, the following problems arise while wrapping information sources:

- The information contributing to the answer for a request is usually distributed over several Web pages. Thus, the wrapper has to navigate the source, collect the required information from the pages, and present only the final result to the user. Interaction with the user must be reduced to a minimum.
- Today more and more commercial search-centers like the Fachinformationszentrum Karlsruhe appear on the literature market. They charge a user not only for the delivery of documents, but also for searching their database. Their search fees are typically based on processing time and result size and are thus hard to predict. The search has to be stopped when a user given limit is reached, eventually with an empty result. To avoid this, a pre-calculation of costs based on meta data and data about previous requests is desirable.
- The functionality provided by sources varies considerably. A uniform wrapper for all information sources would have to cover login-handling, security, payment, registration, metadata collection or result formatting.
- The web pages change their layout frequently. A wrapper accessing a commercial database must check for such changes and make sure it does not spend money on queries whose result pages it is unable to parse.
- Another problem is the wrapper construction. E.g., a university library, which usually has a lot of sources included in their search system, must be able to generate new and modify existing wrappers quickly.

After presenting some other projects in this subject area, we will show our approach to wrapping and wrapper construction for information sources in an electronic commerce environment. Of course these wrappers could also be used for conventional, non-profit information sources.

2 Related Work

Querying Web sources and retrieving data from semistructured and structured Web sources has become more and more important and receives attention in the database literature (see [4] for a survey).

Some projects cover the extraction of information from HTML pages [8, 10, 7]. Most of these use their
own language for defining patterns which must be pro-
grammed by hand. Generating wrappers with these
languages requires substantial programming skills.
The ability to navigate through a sequence of pages is
rarely provided, although this is mandatory for sources
offering literature search in the electronic commerce
environment.

A few projects offer generators to create wrappers
semi-automatically [5, 11]. However, usually these
generators are basically parser generators for HTML
pages. The ability to interact with HTML forms and
to navigate the site intelligently is missing.

Some projects like [1, 6, 3] offer a language or a model
for site navigation. But these languages demand high
programming skills or a detailed knowledge of the site
and its structure. Usually, a librarian would not be able
to use these tools to generate a wrapper.

None of the wrappers mentioned is capable of recog-
nizing changes to the HTML pages and to react in an
adequate manner. However, if a user is paying for the
information, he/she requires correct information.

Finally, existing wrappers lack functionality that is es-
sential in an electronic commerce environment, e.g.
functionality for cost estimation or secure transactions.

In the next part we present our wrapper generator and
our concepts for a flexible wrapping of different in-
formation sources in the electronic commerce envi-
ronment [9]. The main idea is to facilitate an easy
and individual generation of wrappers for every type
of information source by a ‘wrapping by example’ ap-
proach. We also briefly discuss the generated wrap-
er and, in particular how changes in HTML pages are
recognized and the action taken.

3 The Wrappers

The UniCats wrappers are based upon the HTML in-
terface of an information provider. Thus, a wrapper
can be deployed by anybody without requiring access
to internal source interfaces or physical installation at
the source. We also chose XML (eXtensible Markup
Language) as the exchange data model. That is to say,
communication with the wrapper is done by exchang-
ing XML documents. Thus, every search agent or pro-
gram can communicate with our wrapper, assuming
that the interface definition is known.

Our wrappers have to deal with both, conventional
providers such as university libraries and commercial
providers in an electronic commerce environment. To
cover the whole spectrum of conceivable functionality,
we chose a modular architecture: each module has a specific task, and during wrapper generation a
module is inserted if its functionality is needed. We
distinguish two sorts of modules: basic and supple-
mentary modules. Basic modules are required in every
wrapper. They guarantee a minimal functionality that
is sufficient for non-commercial information sources.
Functionality which is required only in the electronic
commerce environment, e.g., cost control and admin-
istration, cost-optimal planning of a search, guaranteeing secure data transmission, or login handling are
provided by supplementary modules.

Figure 1. The wrapper architecture

In Figure 1, the architecture of a UniCats wrapper for
a source in an electronic commerce environment is
shown. Query processing begins when the so-called
coordination module of the wrapper receives a cus-
tomers search request. Then a security level is negoti-
ated and afterwards the request is validated in the so-
called validation module. During this validation pro-
cess, the syntax of the request will be automatically
corrected if required, using meta-information about
the source which has been collected during wrapper
generation. The corrected request is given to a plan-
ner, which tries to construct a plan for a cost-optimal
execution. In doing so, the planner consults a navi-
gation graph. This graph includes all HTML pages
of the providers site with search-relevant information
and links between these pages. Additional information
such as costs or login demands are also stored in this
graph. The result of the planning process is a detailed
execution plan, which includes all steps that must be
performed to get the required result.

When the plan is executed, the results of the search
are usually presented as a series of dynamically gen-
erated HTML pages. Of course, after requesting each
result page the number of results can hardly be fore-
seen. Thus, the planner monitors the costs and, if
necessary, re-calculates and adapts the plan. If a user
given cost limit would be reached, the execution im-
nediately stops, and the user will be asked to increase
the limit or cancel the request.

Often, the user has already to pay for the initial search
request. To avoid an unnecessary request, the wrap-
per collects information from previous requests. With
these data the wrapper tries to make a forecast of the expected number of results and the overall costs without actually sending the request to the provider. So, the user can decide to cancel the request without spending money, e.g., if a request would probably deliver too many documents. Finally, the converter translates the planned request into the syntax of the source, submits the request, transforms the result back and recalculates the plan.

The wrapper shown in Figure 1 was built with the help of predefined modules. The only information which has to be inserted by an administrator is the rules for extracting information out of a page and metadata about the source. This metadata includes the available attributes, cost structure, number and kind of documents, etc. For an easy wrapper generation we apply the principle ‘wrapping by example’. The administrator opens an HTML page with the generator he sees a textual representation with all information on the page. Then an attribute, e.g. the authors name could be selected by clicking in the text and the generator automatically constructs the rule for accessing all authors names on that page. These rules are added in an template file for a later use wrapping that page.

When all attributes on a page are specified, the administrator follows one link on that page like in a normal navigation and could now specify the next page. Some meta information is collected automatically, e.g., the attribute names on the page or a change of servers. Some information requires an entry by the administrator, for example, cost information. So, the generation process can be done even by laymen with no programming skills because it resembles only a navigation and the insertion of additionally information.

In the electronic commerce environment the recognition of changes in the HTML pages is very important. A wrapper has to be adapted immediately, because customers dont like to pay for information they have no use for. Thus, the wrapper always checks the correspondence of the stored and the received structure during a request. Detecting a difference, the page is marked in the navigation graph and the wrapper tries to find an alternative way to get the requested information or refuses the request. The administrator can load the old wrapper into the generator, see the differences and adapt the old wrapper to the new page structure. This is a very fast and easy way to update a wrapper, because only the parts with changes must be replaced, the rest can be re-used.

In this paper we presented a wrapper with a corresponding generator which enables even laymen generating their own wrappers semi-automatically. These wrappers are kept flexible and not restricted to the search of literature, but foreseen especially for an employment in an electronic commerce environment. In the future it is planned to expand the wrapper with different search strategies and optimized algorithms for an query pre-calculation.

References


