Abstract: The major difficulties related to the use of the web interfaces available for extracting data from biomolecular databanks are the HTML format of the output results and the very limited queries that one can submit. This last is an important problem when the user needs to submit many queries regarding a group of genes to various web biomolecular databanks with different interfaces and different structure of the output results. A web wrapper is a software component that assists the human user in such a process by automatically processing HTML pages to extract the data of interest and aggregate data from different sources in a common format. In this paper we analyze some freeware wrapper generation tools. Our analysis focuses on a particularly challenging task for web wrappers: mining and aggregating information of interest for numerous genes or proteins from many distinct biomolecular databanks.

Introduction

New high-throughput technologies in biology - such as DNA microarrays - are generating massive data sets describing, in terms of expression profiles, the biological function of thousands of genes at once. Numerous public web-based databanks describing individual genes and their encoded protein products use heterogeneous web interfaces based on dynamic web pages. Unfortunately, these systems have been created to collect and present data organized for individual nucleotide or amino acidic sequences [1,2]. Thus it is possible to query only one gene at a time. For this reason most valuable web databanks poorly adapt to be used in biological interpretative analyses and knowledge discovery from high-throughput expression profile results. To perform a comparative evaluation of multiple characteristics of many nucleotide and amino acidic sequences at once requires the support of automated tools. Furthermore, if we want to search the same information in different web sites, we will have to deal with different visualization of the same data. We believe that the task of extracting all informations regarding a group of genes or proteins from many different web databanks is general enough to represent most of the difficulties typical of such domain.

A software component that converts data and queries from one model to another is called a wrapper. A good wrapper should be able to convert information implicitly stored as an HTML document into information explicitly stored as a data-structure for further processing. As the structure of a dynamic page can change slightly from one query to another (e.g. tables with extra rows or missing fields) a wrapper should be able to locate and extract the data of interest inside the HTML page. Unfortunately, this is not a simple task, as HTML was designed for visualization purposes. In fact, HTML tags describe how the data will be visualized, not its semantics. Thus, Human-readability and machine-readability are often not equivalent.

In this paper we review the main characteristics of state-of-the-art wrapper tools [3]. Then we will discuss how some available wrapper tools satisfy such characteristics and what performance we could expect from them in the biomedical research domain.

Materials and Methods

Wrapper Generation

The observation that writing a wrapper by hand is tedious and error-prone led to the creation of Wrapper Generation software tools. A Wrapper Generation Toolkit hides all repetitive tasks in the creation of a wrapper. Users can take advantage of a Graphical User Interface to build a wrapper for a specific Internet source. A Wrapper Generation Toolkit holds all information necessary to generate and run the wrapper in a wrapper description file. In addition some tools can export the wrapper as a Java class file.

Case Study

We started from the point of view of a small research laboratory, or even an individual investigator, having low funding level. The goal is to perform biological interpretative analyses and knowledge discovery from high-throughput expression profile results. Our case study was extracting all information regarding a group of genes from different web-based biomolecular databanks. This task to us involves most
of the significant difficulties encountered when mining data from data-rich HTML sources. Such difficulties are mainly consequent to variations in the page structure. Depending on the gene or amino acidic sequence requested, quantity of available information may vary. One of the most frequent cases is that of tables having extra rows or missing fields from one query to another. For this reason the wrapper needs to implement a strategy to locate correctly information of interest. In the next sections we will explain some of these strategies.

Wrapper Grid Categories

In this section we introduce a grid of features to evaluate wrapper tools for biomedical applications. We start explaining some simple categories from our grid. First of all, since biomedical researchers are not supposed to have advanced computer knowledge, a good tool must have a clear and complete Graphical User Interface with help and tutorial. Secondly, the tool should not require writing code to generate the wrapper. Another feature we decided to be indispensable is the possibility to perform numerous batch extractions on different queries without user participation. This is crucial because HTML extractions for many different genes require time and a batch component could even update easily data extracted in previous extractions.

Here we list our Grid Categories (we already explained the first three, the other will be described in the next section):

- Have a clear and complete GUI plus sample data and tutorials.
- Do not require code writing.
- To provide a batch mode to perform and update numerous extractions.
- Use HTML-based representation.
- Implement Wrapper Induction heuristics.
- Hold a model of the data to be extracted (we will call this method Data Model approach).

Explanation of Grid Categories

The tasks a wrapper generation tool must pass through before he can generate the wrapper are:

1. Input one or more example from the user through a GUI.
2. Generate extraction rules to extract data from the example.
3. Refine these rules to be more accurate or more flexible when the page structure changes.
4. After the wrapper is generated it can be used to extract data from HTML pages, therefore the wrapper has to:
5. Perform the extraction
6. Store the extracted data in a suitable format for further analysis (e.g. XML, spreadsheet file, relational tables).

HTML-based wrappers parse HTML input and immediately build a tree representation (based on W3C’s Document Object Model, see http://www.w3.org/DOM/) out of the page. This representation is used to navigate the tree and locate the information we need to extract. HTML-based tools often feature a declarative notation designed to describe the navigation along the tree to reach the data of interest.

Wrapper Induction [4-6] algorithms try to introduce some “intelligence” in the extraction rule generation task. The system, given a sufficient set of examples, automatically generates (“induce”) rules that cover all examples, and hopefully all cases similar but not equal to the examples. The target document is represented as a set of tokens (characters, group of characters or HTML tags). Tokens surrounding the data of interest are used as anchors. A side effect when using Wrapper Induction techniques is the system requires a set of training examples, as one is often not sufficient.

We say a wrapper is “Data Model Oriented” when it holds an internal representation of the abstract structure of the data to extract. The system stores the hierarchy of the tuples and attributes that compose a complex data, the type of each attribute, the number of fields in a table and so on. The Data Model must be as detailed as possible, but if it becomes too much detailed the wrapper can lose flexibility (e.g. it cannot find data if it doesn’t have all the attributes expected). Heuristic algorithms used come from machine learning or pattern matching techniques.

Evaluated Wrapper Tools

Wrapper tools are quite common but, according to our domain of application, we chose to focus only on GUI freeware tools [7-10], suitable even for low-budget research laboratories. In practice freeware tools are quite rare, thus two of the tools we found are part of commercial tools with time-limited license or demo versions with only the essential modules. We performed our testing with the freely available modules. XWRAP and GeneWebEx [8,10] are totally free. The first three tools are general-purpose web wrapper tools, the last one was specifically designed to mine data from web-based biomolecular databanks.

Results and Discussion

Wrapper Tool Description

W4F (http://cheops.cis.upenn.edu/W4F/) works on DOM representation of the page. It comes with a GUI where the user can select relevant data in a page. The system generates automatically the rules code for each selection. Basing on extraction rules, the system can output a working wrapper as a Java class file. W4F is composed of three modular layers, one of which is the extraction section. The others are a retrieval section, where the user can specify details concerning HTML
page retrieval. In this section it is possible to express the target page’s URL in a parametric way, suitable for submitting different queries on the same page. The third section consists of mapping rules. Mapping rules specify the binding between the proprietary internal data representation and the output notation. Default mappings are pre-defined towards Java base types and XML. The advanced user can specify his own mappings towards any external representation.

**XWRAP** is a wrapper generation service available at [http://www.cc.gatech.edu/projects/disl/XWRAPElite/](http://www.cc.gatech.edu/projects/disl/XWRAPElite/). After the user has logged on to the system a GUI wizard drives him through a number of steps for wrapper generation. First the specified URL is pre-processed to clean HTML code from incorrect tags, then a tree representation of the page is built. The user highlights the data he wants to retrieve directly in the GUI. The system shows what extraction objects are found at each step and each preliminary result can be refined through various feedback interactions with the user. The description of the data structure is stored in an internal representation based on XML templates to facilitate the generation of the wrapper code. As a final result, XWRAP generates the wrapper as a Java class file that can be downloaded from the site. Extraction results are XML formatted.

**DEByE** ([http://www.lbd.dcc.ufmg.br/~debye/](http://www.lbd.dcc.ufmg.br/~debye/)) exploits a full Data-Model approach. As in the other considered tools the interaction with the user is guided via-GUI. The Data-Model representation used describes the information to extract with a relational style. DEByE saves a wrapper description file to perform extractions. The results of extraction are produced in an XML-based format suitable for data exchange between applications.

**GeneWebEx** stands for “Gene Web Extractor” ([http://www.medinfopoli.polimi.it/GeneWebEx/](http://www.medinfopoli.polimi.it/GeneWebEx/)). It was developed specifically for mining data from biological databanks, but we are sure that it is suitable for other fields of application beside the medical area. The system renders the HTML page into a DOM-like tree representation. Extraction templates describe navigation along the tree to reach relevant data, as in all HTML-based tools. Similarly to the DEByE tool, GeneWebEx performs directly the extraction according to the template used. It is possible to store the results of the extraction into text Excel files or directly in a local relational database with built-in queries for further analysis and data mining. An updating agent module retrieves in batch mode information presenting high temporal variability, maintaining local data synchronized to those in the original databanks.

**Wrapper Tool Evaluation**

We tested the tools on a set of genomic databases ([1,2]) sample pages and we encountered many difficulties confirming how complex is this domain. Genomic annotations pages present a complex structure (e.g. nested tables, data shown as GIF images) and a high variability, because some informations are not available for every gene (e.g. related diseases). None of the tested tools showed total reliability in case of deep variations in the page structure. GeneWebEx had the best performance, but this is natural because it was designed for this specific purpose. The other tools did only extract a part of significant information, but in some cases showed better flexibility and “intelligence” in locating data automatically.

Results of our evaluation according to our grid categories are summarized in Table 1.

<table>
<thead>
<tr>
<th>Tool</th>
<th>W4F</th>
<th>XWRAP</th>
<th>DEByE</th>
<th>Gene WebEx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete GUI</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>No Code Writing</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>HTML Aware</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Data Model Oriented</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Wrapper Induction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch Extractions</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

We can say W4F is HTML-based as rules are generated declaratively, wrapper induction is absent. GUI is incomplete as it is designed only to help writing code. The system does not handle any change in the target pages, the user must handle such changes writing better extraction rules. A Data Model is not implemented, as extraction rules specify simply DOM tree paths. Once the system has found a keyword it fetches data near to it without making any assumptions on its structure. A module for batch extractions is absent, but a programming-skilled user could write it taking advantage of the Java wrapper class generated by the tool.

XWRAP uses both a HTML-based representation and some powerful Wrapper Induction heuristics. As a consequence of using heuristics for inducing rules it requires more than one example for each extraction, and more than one feedback from the user via GUI. Data Model approach is not implemented, but XWRAP shows some “intelligence” in locating significant data into HTML code. As a ready-to-work tool however it is still incomplete because it outputs the wrapper as a Java class file. Writing manually additional modules is required to perform batch extractions and further analysis on extracted data.

DEByE’s heuristics do not require code writing, but it is difficult to perform correct extractions and to
correct extraction errors, as only few options are selectable via GUI. This tool is purely Data-Model oriented because the HTML structure of the page is not considered. The tool works even on other page formats like XML and LaTeX. However, it is not easy to generate good wrappers due to the obscure GUI and the lack of a tutorial with examples. A module for batch extractions is not present.

GeneWebEx’s main features are a simple GUI and its completeness, as it includes all-in-one the modules necessary for extraction and data-mining of biomedical data. GeneWebEx takes advantage of its simple HTML-based approach, but extraction errors can occur when the pattern between the anchor and the data to be extracted changes respect to the template used. In this case the user can easily discover what went wrong by viewing simple log files. Then he can try to extend the template to handle such exceptions. The tool does not require to write code. Storing results in a local database permits to perform data mining queries directly on extracted data.

In Table 2 we summarize some extra features of examined tools not included in our grid categories. These features are significant when the wrapper tool needs to be integrated with other applications. The first is the possibility to generate the wrapper as a java class file. The second is the output format of extracted data.

### Table 2: Wrapper Tool Interfaces

<table>
<thead>
<tr>
<th>Tool</th>
<th>W4F Generates Java code</th>
<th>XWRAP Output Format</th>
<th>DEByE Output Format</th>
<th>GeneWebEx Output Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>W4F</td>
<td>Yes</td>
<td>XML</td>
<td>No</td>
<td>XLS, Relational DB</td>
</tr>
<tr>
<td>XWRAP</td>
<td>Yes</td>
<td>XML</td>
<td>No</td>
<td>XML</td>
</tr>
<tr>
<td>DEByE</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

### Conclusions

We found general purpose wrapper tools are not directly applicable in the biomedical domain, as they miss some important features, such as a complete GUI, a batch module and the possibility to generate a wrapper without code writing. Furthermore, our tests showed that we can expect few flexibility from web wrappers operating in such a difficult domain. Thus, it is essential to provide a clear interface dedicated to discovery and correction of extraction error. A preliminary step may be the implementation of automatic queries on log files to report efficiently “data not found” errors. Incorrect HTML tags also cause frequently extraction errors. Therefore we can expect better performance if a module parses the HTML page to correct syntax errors before page processing. Finally we found that performance of web wrappers improves when Wrapper Induction and Data Model approaches are implemented.

### References


