A Brief Survey of Web Data Extraction Tools[†]

Alberto H. F. Laender Berthier A. Ribeiro-Neto
Altigran S. da Silva[‡] Juliana S. Teixeira
Department of Computer Science
Federal University of Minas Gerais
31270-901 Belo Horizonte MG Brazil
{laender,berthier,alti,juliana}@dcc.ufmg.br

ABSTRACT

In the last few years, several works in the literature have addressed the problem of data extraction from Web pages. The importance of this problem derives from the fact that, once extracted, the data can be handled in a way similar to instances of a traditional database. The approaches proposed in the literature to address the problem of Web data extraction use techniques borrowed from areas such as natural language processing, languages and grammars, machine learning, information retrieval, databases, and ontologies. As a consequence, they present very distinct features and capabilities which make a direct comparison difficult to be done. In this paper, we propose a taxonomy for characterizing Web data extraction tools, briefly survey major Web data extraction tools described in the literature, and provide a qualitative analysis of them. Hopefully, this work will stimulate other studies aimed at a more comprehensive analysis of data extraction approaches and tools for Web data.

1. INTRODUCTION

With the explosion of the World Wide Web, a wealth of data on many different subjects has become available online. This has opened the opportunity for users to benefit from the available data in many interesting ways [7]. Usually, users retrieve Web data by browsing and keyword searching, which are intuitive forms of accessing data on the Web. However, these search strategies present several limitations. Browsing is not suitable for locating particular items of data, because following links is tedious and it is easy to get lost. Keyword searching is sometimes more efficient than browsing, but often returns vast amounts of data, far beyond what the user can handle. As a result, in spite of being publicly and readily available, Web data can hardly be properly queried or manipulated as done, for instance, in traditional databases.

For handling Web data more effectively, some researchers have resorted to ideas taken from the database area [13]. Databases, however, require structured data and, therefore, traditional database techniques cannot be directly applied to Web data. The advent of XML [6] as a standard for structuring data available on the Web has brought some light to this problem, but this technology *per se* does not provide a trivial solution for properly manipulating existing Web data. Indeed, the volume of unstructured or semistructured data available on the Web is enormous and is still increasing. Thus, to address this problem, a possible strategy is to extract data from Web sources to populate databases for further handling.

The traditional approach for extracting data from Web sources is to write specialized programs, called *wrappers*, that identify data of interest and map them to some suitable format as, for instance, XML or relational tables. The most challenging aspect of wrappers is that they must be able to recognize the data of interest among many other uninteresting pieces of text (e.g., markup tags, inline code, navigation hints, etc.). These data might have a flat structure, but might also be complex and present an implicit multi-level hierarchical structure, that is often non-rigid. This means that the data may exhibit structural variations that must be tolerated and treated accordingly.

Developing wrappers manually has many well known shortcomings, mainly due to the difficulty in writing and maintaining them. Recently, many tools have been proposed to better address the issue of generating wrappers for Web data extraction [3, 4, 8, 9, 10, 11, 14, 16, 18, 20, 23, 28, 30, 31]. Such tools are based on several distinct techniques such as declarative languages [4, 9, 16], HTML structure analysis [10, 23, 31], natural language processing [14, 28, 32], machine learning [8, 18, 20], data modeling [3, 30], and ontologies [11].

The problem of generating a wrapper for Web data extraction can be stated as follows. Given a Web page S containing a set of implicit objects, determine a mapping W that populates a data repository R with the objects in S. The mapping W must also be capable of recognizing and extracting data from any other page S' *similar* to S. We use the term *similar* in a very empirical sense, meaning pages provided by a same site or Web service, such as pages of a same Web bookstore. In this context, a wrapper is a program that executes the mapping W. A common goal of all

[†] This work was partially supported by Project SIAM (MCT/CNPq/PRONEX grant number 00418.00/00) and by the authors' individual grants from CNPq and CAPES.

[‡] On leave from the University of Amazonas, Brazil.

wrapper generation tools is to generate wrappers that are highly accurate and robust, while demanding as little effort as possible from the wrapper developers. As we shall see, in practice, this imposes an important trade-off between the degree of automation of a tool and the flexibility of the wrappers generated by it.

As more and more tools for Web data extraction continue to appear, the need for the analysis of their capabilities and features arises. In this paper we briefly survey some of the tools described in the literature, also discussing how some features that we regard as most important for Web data extraction are supported by each tool.

A pioneer initiative for comparing tools and techniques for Web data extraction is the RISE Web site [26]. In this site, experimental results performed using STALKER [28], WHISK [32], SRV [14], WIEN [20], and RAPIER [8], which are tools based on either machine learning or natural language processing, have been made available. These quantitative results were submitted by the authors of the tools themselves. Unfortunately, only in some few cases results generated by distinct tools are directly comparable. In [27], Muslea compares and contrasts various types of extraction patterns that are generated by different types of machine learning algorithms. However, a more comprehensive analysis of data extraction approaches and tools for Web data has not yet been carried out, mainly due to the difficulty to establish a general framework for such a comparison. The present paper is a first attempt in this direction.

The paper is organized as follows. In Section 2 we introduce a taxonomy for characterizing Web data extraction tools. Section 3 presents an overview of major data extraction tools found in the literature. Following, in Section 4, we present a qualitative analysis of these tools. Finally, Section 5 presents our conclusions.

2. A TAXONOMY FOR CHARACTERIZING WEB DATA EXTRACTION TOOLS

This section presents a taxonomy for grouping the various tools we have studied. This taxonomy is based on the main technique used by each tool to generate a wrapper, what led us to the following groups of tools: *Languages for Wrapper Development*, *HTML-aware Tools*, *NLP-based Tools*, *Wrapper Induction Tools*, *Modeling-based Tools*, and *Ontology-based Tools*.While such a taxonomy is useful for didactic purposes, it must not be taken as the only possibility. In fact, there are cases where a same tool could fit in two or more of the identified groups. However, the proposed taxonomy is helpful as a guide for properly understanding the existing approaches to Web data extraction and to assess the suitability of the covered tools for specific applications. In what follows, we describe the main characteristics of the tools belonging to each group.

Languages for Wrapper Development.

One of the first initiatives for addressing the problem of wrapper generation was the development of languages specially designed to assist users in constructing wrappers. Such languages were proposed as alternatives to general purpose languages such as Perl and Java, which were prevalent so far for this task. Some of the best known tools that adopt this approach are Minerva [9], TSIMMIS [17], and Web-OQL [4]. Other representative tools of this approach are FLORID [24] and Jedi [19], but these tools are not covered in this paper.

HTML-aware Tools.

We group here tools that rely on inherent structural features of HTML documents for accomplishing data extraction. Before performing the extraction process, these tools turn the document into a parsing tree, a representation that reflects its HTML tag hierarchy. Following, extraction rules are generated either semi-automatically or automatically and applied to the tree. Some representative tools based on such an approach are W4F [31], XWRAP [23], and RoadRunner[10]. Another tool that can be regard as HTML-aware is Lixto [5], but we will not discuss it in this paper.

NLP-based Tools.

Natural language processing (NLP) techniques have been used by several tools to learn extraction rules for extracting relevant data existing in natural language documents. These tools usually apply techniques such as filtering, part-of-speech tagging, and lexical semantic tagging to build relationship between phrases and sentences elements, so that extraction rules can be derived. Such rules are based on syntactic and semantic constraints that help to identify the relevant information within a document. The NLP-based tools are usually more suitable for Web pages consisting of free text, possibly in telegraphic style, such as job listings, apartment rental advertisements, seminar announcements, etc. Representative tools based on such an approach are RAPIER [8], SRV [14], and WHISK [32].

Wrapper Induction Tools.

The wrapper induction tools generate delimiter-based extraction rules derived from a given set of training examples. The main distinction between these tools and those based on NLP is that they do not rely on linguistic constraints, but rather in formatting features that implicitly delineate the structure of the pieces of data found. This makes such tools more suitable for HTML documents than the previous ones. Tools such as WIEN [20], SoftMealy [18], and STALKER [28] are representative of this approach.

Modeling-based Tools.

This category includes tools that, given a target structure for objects of interest, try to locate in Web pages portions of data that implicitly conform to that structure. The structure is provided according to a set of modeling primitives (e.g, tuples, lists, etc.) that conform to an underling data model. Following, algorithms similar to those used by the wrapper induction tools identify objects with the given structure in the target pages. Tools that adopt this approach are NoDoSE [3] and DEByE [21, 30].

Ontology-based Tools.

All approaches described previously rely on the structure of presentation features of the data within a document to generate rules or patterns to perform extraction. However, extraction can be accomplished by relying directly on the data. Given a specific domain application, an *ontology* can be used to locate constants present in the page and to construct objects with them. The most representative ontology-based tool is the one developed by the Brigham Young University Data Extraction Group [11].

3. OVERVIEW OF WEB DATA EXTRACTION TOOLS

In this section we overview the Web data extraction tools we have studied. We notice that the list of tools covered here must not be regarded as complete. Although we have tried to cover the most representative tools that have appeared in the recent literature, our study is not exhaustive. The presentation of the tools follows the taxonomy introduced in Section 2.

3.1 Languages for Wrapper Development

Minerva – An important component of the Araneus system [25] is Minerva [9], a formalism for the development of wrappers. Minerva combines a declarative grammar-based approach with features typical of procedural programming languages. The grammar used by Minerva is defined in the EBNF style: for each document, a set of *productions* is defined; each production defines the structure of a non-terminal symbol of the grammar, in terms of terminal symbols and other non-terminals. Minerva is complemented by a language for searching and restructuring documents called Editor, which provides basic operations found in text editors. For dealing with irregularities commonly found in Web data, Minerva provides an explicit procedural mechanism for handling exceptions inside of the grammar parser. For each production of the grammar, it is possible to add an *exception clause*, containing a piece of Editor code. Whenever the parsing of that production fails, an exception is raised and the corresponding exception code is executed.

TSIMMIS – Among other components for semistructured data management, TSIMMIS [17] includes wrappers that can be configured through specification files written by the user [16]. Specification files are composed by a sequence of commands that define extraction steps. Each command is of the form [*variables, source, pattern*] where *variables* represents a set of variables that hold the extraction results, *source* specifies the input document to be considered (e.g., a Web page), and *pattern* allows matching the data of interest within the source. The data stored in the variables can be used as input for subsequent commands. An extractor based on a specification file parses an HTML page to locate the interesting data and extract them. After the last command is executed, the set of variables holds the extracted data. Although there is no language for wrapper development formally defined for TSIMMIS, we included it in this survey because of its historical importance and pioneering.

Web-OQL – Originally aimed at performing SQL-like queries over the Web, Web-OQL [4] is a declarative query language that is capable of locating selected pieces of data in HTML pages. For this, a generic HTML wrapper parses a page given as input and produces as result an abstract HTML syntax tree, called a *hypertree*, representing the document. Using the syntax of the language, it is possible to write queries that locate data of interest in the *hypertree* and then output these data in a suitable format (e.g., tables). This is how Web data extraction can be accomplished using Web-OQL. Navigation through *hypertrees* is also supported.

3.2 HTML-aware Tools

W4F (World Wide Web Wrapper Factory) – W4F [31] is a toolkit for building wrappers. W4F divides the wrapper development process in three phases: first, the user describes how to access the document, second, he describes what pieces of data to extract, and third, he declares what target structure to use for storing the data extracted. A document is first retrieved from the Web according to one or more retrieval rules. Once retrieved, it is fed to an HTML parser that constructs a parsing tree following the Document Object Model (DOM) [34]. Following, the user can write extraction rules for locating data into the parsing tree. The extracted data is stored using the W4F internal format, called NSL (Nested String List). Finally, NSL structures can be exported to upper-level applications, according to specific mapping rules. The language used by W4F to define extraction rules is called HEL (HTML Extraction Language). An extraction rule is an assignment between a variable name and a path expression. W4F offers a wizard to assist the user in writing extraction rules that are applied to tree nodes to extract data. For a given Web document, the user is presented with the same document annotated with additional information. The user clicks on the pieces of information of interest and the wizard returns a corresponding extraction rule. The wizard cannot deal with collection of items, so if the user is interested in various items of the same type of that one clicked on, conditions must be attached to the path expression to write robust extraction rules.

XWRAP – Another important HTML-aware tool for semi-automatic construction of wrappers is XWRAP [23][1]. The tool features a component library that provides basic building blocks for wrappers, and a user-friendly interface to ease the task of wrapper development. Before accomplishing the extraction process, the tool "cleans up" bad HTML tags and syntactical errors and turns the document into a parsing tree. The tool operates by leading the user through a number of steps, selecting in each step proper components of its library. At the end, XWRAP outputs a wrapper (coded in Java) for a specific source. In the object extraction step, the tool deploys a pre-defined set of data extraction heuristics tailored for HTML pages. The user may try one of six heuristics available to locate data objects of interest. If the user is satisfied with the extraction results, the extraction process may go on. The user can also refine the extraction by restricting or relaxing the number of components per object or by specifying data types for the elements. When the extraction result is satisfactory, the user may enter a tag name for each of the elements extracted and proceed to the wrapper code generation step.

RoadRunner – A recent tool that further explores the inherent features of HTML documents to automatically generate wrappers is RoadRunner [10]. It works by comparing the HTML structure of two (or more) given sample pages belonging to a same "page class", generating as a result a schema for the data contained in the pages. From this schema, a grammar is inferred which is capable of recognizing instances of the attributes identified for this schema

[1] In this paper, we cover only the XWRAP Elite version, which is available for use at the URL http://www.cc.gatech.edu/projects/disl/XWRAPElite/

in the sample pages (or in pages of the same "class"). To accurately capture all possible structural variations occurring on pages of a same page class, it is possible to provide more than two sample pages. All the extraction process is based on an algorithm that compares the tag structure of the sample pages and generates regular expressions that handle structural mismatches found between the two structures. In this way, the algorithm discovers structural features such as tuples, lists, and variations. It should be noted that the process is fully automatic and no user intervention is requested, a feature that is unique to RoadRunner.

3.3 NLP-based Tools

RAPIER (Robust Automated Production of Information Extraction Rules) — RAPIER is a tool aimed at extracting data from free text. It takes as input a document and a filled template indicating the data to be extracted. This template is used to learn data extraction patterns to extract data for populating its slots. The learning algorithm incorporates techniques from several inductive logic programming tools and learns unbounded patterns that include constraints on the words and part-of-speech tags surrounding the filler data. These patterns consist of three distinct slots: the *Pre-*, *Post-*, and *Filler*. The formers play the role of left and right delimiters, while the later describes the structure of the data to be extracted. RAPIER extracts a single record from each document taken as input, and is therefore termed to be "single-slot" [32].

SRV – Based on a given set of training examples, SRV [14] is a tool for learning extraction rules for text documents. It relies on a set of *token-oriented features* that can be either *simple* or *relational*. A simple feature is a function mapping a token to some discrete value, for example, word, punctuation, or numeric. A relational feature maps a token to another token, for instance, prev-token or next-token. The learning of rules consists in identifying and generalizing the features found in the training examples. The existence of a number of HTML-specific features (such as in-title or after-p) in its default feature set, makes SRV able to extract data from HTML documents. SRV is also a "single-slot" tool.

WHISK – Another tool for data extraction from text documents is WHISK [32]. In this tool, a set of extraction rules is induced from a given set of training example documents. Beginning with an empty set of rules, at each iteration it selects and presents to the user a batch of instances to be tagged. The user uses a graphical interface to add a tag for each attribute of interest to be extracted from the instance. Then, WHISK uses the tagged instances to create rules and also to test the accuracy of the proposed rules. These rules are based on a form of regular expression patterns that identify the context of relevant phrases and the exact delimiters of those phrases. WHISK is "multi-slot", i.e., it is capable of extracting several records from a document.

3.4 Wrapper Induction Tools

WIEN – A pioneer wrapper induction tool is WIEN [20], which takes as input a set of pages where data of interest is labeled to serve as examples, and returns, as a result, a wrapper that is consistent with each labeled page. The pages are assumed to have a pre-defined structure and specific induction heuristics are used to generate specific wrappers. For instance, if the pages have an *HLRT* structure (i.e., pages have a head, a body containing flat tuples of data delineated by a left and a right component to be extracted, and then a tail), an *HLRT wrapper* is generated. Wrappers generated by WIEN do not deal with nested structures or with variations typical of semistructured data.

SoftMealy – Similar to WIEN, SoftMealy [18] is a wrapper induction tool that generates extraction rules expressed using a special kind of automata called *finite-state transducers* (FST). An FST consists of input/output alphabets, states, and edges. To deal with structural variations, each state of the FST may have multiple outgoing edges. Before extracting data from a document, the wrapper segments an input HTML string into tokens, then the algorithm tries to induce extraction rules based on the context formed by the separators (tokens) of adjacent attributes present in given training examples. The resulting FST takes a sequence of tokens as input and matches the context separators with contextual rules to determine state transitions. An FST is constructed for one tuple type. If there can be many types of tuples in a document, an FST can be built for each type.

STALKER – The wrapper induction techniques used in WIEN and SoftMealy are further developed in STALKER [28], a tool that can deal with hierarchical data extraction. The inputs to STALKER are: (1) a set of training examples in the form of a sequence of tokens representing the surrounding of the data to be extracted; (2) a description of the pages structure, called an *Embedded Catalog Tree* (*ECT*). STALKER generates an extraction rule that covers as many as possible of the given examples. While uncovered examples exist, it generates a new disjunctive rule. When all positive examples are covered, STALKER returns the solution, that consists of a set of disjunctive rules. Using the *ECT*, STALKER can deal with nesting hierarchical objects.

3.5 Modeling-based Tools

NoDoSE (Northwestern Document Structure Extractor) – NoDoSE [3] is an interactive tool for semi-automatically determining the structure of documents that contain semistructured information and then extracting their data. Using a graphical user interface, the user hierarchically decomposes the document, outlining its interesting regions and describing their semantics. The decomposition process of a document occurs in levels. For each level of decomposition, the user builds an object with a complex structure, and then decomposes it in other objects with a more simple structure. After the user has "taught" the tool how to construct some objects, he can let NoDoSE to learn how to identify other objects in the document. This is accomplished by a *mining* component that attempts to infer the grammar of the document from objects constructed by the user. In its current version, NoDoSE features mining components for plain text and for HTML pages.

DEByE (Data Extraction By Example) – DEByE [21, 30] is an interactive tool that receives as input a set of example objects taken from a sample Web page and generates extraction patterns that allow extracting new objects from other similar pages (e.g., pages from a same Web Site). DEByE features a GUI that allows the user to assemble nested tables (with possible variations in structure) using pieces of data taken from the sample page. The tables assembled are examples of the objects to be identified on the target pages. From these examples, DEByE generates *object extraction patterns (OEP)* that indicate the structure and the textual surroundings of the objects to be extracted. These OEP are then fed to a *bottom-up extraction algorithm* that takes a target page as input, identifies atomic values in this page, and assembles complex objects using the structure of the OEP as a guide.

3.6 Ontology-based Tools

This approach is mainly represented by the work of the Data Extraction Group [11] at Brigham Young University (BYU). In their tool, ontologies are previously constructed to describe the data of interest, including relationships, lexical appearance, and context keywords. By parsing this ontology, the tool can automatically produce a database by recognizing and extracting data present in documents or pages given as input. Prior to the application of the ontology, the tool requires the application of an automatic procedure to extract chunks of text containing data "items" (or records) of interest [12].

To work properly, this tool requires a careful construction of an ontology, a task that must be done manually by an expert in the domain of the ontology. On the positive side, if the ontology is representative enough, the extraction is fully automated. Furthermore, wrappers generated according to such an approach are inherently resilient (i.e., they continue to work properly even if the formatting features of the source pages change) and adaptable (i.e., they work for pages from many distinct sources belonging to a same application domain). Indeed, these features are unique to this approach. For convenience, in the remainder of the paper we will refer to this tool as the BYU tool.

As another example of an ontology-based tool for data extraction, we could cite X-tract [1], a tool for extracting data from botanical textual descriptions. However, as this tool applies only to a very specific domain, we will not discuss it further in this paper.

4. QUALITATIVE ANALYSIS

In this section, we analyze how the studied tools support some features that we regard as most important for data extraction. We address the following features: degree of automation, support for complex objects, page contents, ease of use, XML output, support for non-HTML sources, and resilience and adaptiveness.

4.1 Degree of Automation

A very important feature of any data extraction tool is its degree of automation. This is related to the amount of work left to the user during the process of generating a wrapper for extracting Web data.

Regarding the degree of automation, the approaches based on languages for wrapper generation still require the writing of code, but provide some features, not available in general purpose languages, that ease this task. In tools such as Minerva, TSIMMIS, and Web-OQL, the user must examine the document and find the HTML tags that separate the objects of interest, and then write a program to separate the object regions. In other words, the process of discovering object boundaries is carried out manually.

HTML-aware tools usually provide a higher degree of automation. However, for this automation to be really effective, there must be a very consistent use of HTML tags in the target page. Unfortunately, this is not true for a great portion of Web pages available. In XWRAP, for example, the component library has a number of predefined heuristics to deal with several types of structuring HTML markups (e.g., tables, lists, etc.). By applying such heuristics and asking for feedback from the user, the tool can extract data very efficiently from certain type of pages. W4F uses the HTML Extraction Language (HEL) to define extraction rules. It features an extraction-wizard which can return a canonical path expression for a piece of information selected by the user. As the wizard cannot deal with collection of items, the user who is interested in various items of the same type must manually write extraction rules that generalize the path expressions provided by the wizard. That is, the extraction process is semi-automatic. RoadRunner, as previously discussed, is fully automatic. In particular, the extraction procedure assumes that the target pages were generated from some data source (e.g., a database). Then, several heuristics are used to "reconstruct" the schema of such a data source from the HTML tag hierarchy of the sample pages. This exempts users from supplying a target schema as well as examples of the data to be extracted.

The tools based on NLP, wrapper induction, and modeling are said to be *semi-automatic* because, although the wrapper generation process is fully automatic, the user has to provide examples to guide it.

As already discussed, the BYU tool requires the construction of an ontology to work properly, what should be done manually by an expert in the corresponding domain. After this, if the ontology is representative enough, the extraction is fully automated and can be used for other data sources in the same domain. Indeed, this feature is unique of such an approach. However, the ontology construction usually requires substantial effort for being validated.

4.2 Support for Complex Objects

Most of the data available on the Web implicitly presents a complex structure. Typically, this structure is loose, presenting degrees of variation typical of semistructured data [2]. Further, in many situations, Web data is organized in hierarchies with multiple nesting levels. Thus, wrapper generation tools are expected to deal with such complex objects properly.

The exception mechanism and the Editor language featured in Minerva make it suitable to deal with the variations normally found on Web data. They are used to properly restructure the data of interest, whenever a production of the grammar fails. To represent complex objects, TSIMMIS adopts the Object Exchange Model (OEM) [29]. OEM is a flexible model very suitable for representing semistructured data. Data represented in OEM constitute a graph, with a unique root object at the top and zero or more nested sub-objects. Web-OQL is capable of querying pages with irregular structure. The language, as well as the object model based on *hypertrees* (ordered arc-labeled trees) adopted by it, allows the representation of objects with structural variations and nested levels.

The language used by W4F to define extraction rules, HEL, provides some operators that allow constructing objects with complex structures. For instance, by using the *fork* operator [31], the user can group together in a single structure data that appear in several places. This operator can be used in cascade making it possible to build complex and irregular structures. In all of these cases, it is possible to handle complex objects by writing extraction code to deal with them. XWRAP, on the other hand, can only deal with nesting and variation if they are explicitly defined in the HTML formatting of the source page. It is able to determine the nesting hierarchy of the source page, by identifying top-level HTML structures (e.g., sections, tables) that form the page and internal structures (e.g., columns, rows, subsections). A similar approach is adopted by RoadRunner, that is based on the notion of nested types, which allows representing arbitrarily nested structures composed of lists and tuples.

In SoftMealy, the wrapper is represented as a FST where each state may have multiple outgoing edges. This allows the representation of structural variations in the code of the generated wrapper, making it capable of handling structural variations. SoftMealy, however, does not deal with nested structures. STALKER is more expressive than SoftMealy in this regard, since it uses an *Embedded Catalog Tree* formalism to describe the structure of the data contained in Web pages. This formalism represents the structure of the target page as a tree, where the internal nodes represent complex objects that can be decomposed and the external nodes (leaves) represent atomic data items to be extracted. This makes it able of dealing with nested structures. Structural variations are handled by generating disjunctive rules from the training examples provided by the user.

NoDoSE maintains a tree that maps the structural elements of the document to the text of the file. Each node of the tree represents one of the structural components of the document such as an element of a list or a field in a record. In DEByE, the underlying data model [21, 22] extends the usual notion of nested tables by allowing the representation of variations inside inner levels. Although such a model is not as powerful as XML or OEM, it is expressive enough to represent data presenting a hierarchical structure and structural variations.

RAPIER, SRV, WHISK, and WIEN support neither nesting objects nor objects with structural variations.

4.3 Page Contents

With respect to page contents, there are basically two kinds of pages which wrapper generation tools apply to: those containing *semistructured data* and those containing *semistructured text*. To illustrate, consider the pages presented in Figures 1 and 2, which are examples of pages containing semistructured data and semistructured text, respectively[2]. While pages of the first type feature data items (e.g., names of authors, titles of papers, etc.) implicitly formatted to be recognized individually, pages of the second type bring free text from which data items can only be inferred.

Languages for wrapper development (Minerva and Web-OQL), HTML-aware tools (W4F, XWRAP, and RoadRunner), wrapper induction tools (WIEN, SoftMealy, and STALKER), and modeling-based tools (NoDoSE and DEByE) usually rely on delimiters surrounding data of interest to generate extraction rules. Thus they work better with pages of the first type.

Tools based on natural language processing techniques, such as RAPIER, SRV, and WHISK, are generally more suitable to pages of the second type (e.g., job listings, apartment rental advertisements, etc.), but require that pages containing free text to be annotated by a syntactic analyzer and semantic tagger.

As the BYU tool relies on the presence of recognizable constants and keywords in the target page, it can be applied to both types of pages. Indeed, the authors of this tool present experimental results that corroborate this [11]. Notice, however, that the accuracy of the wrapper generated, for both types of page, depends on how representative is the ontology for the domain to which the pages belong.

4.4 Ease of Use

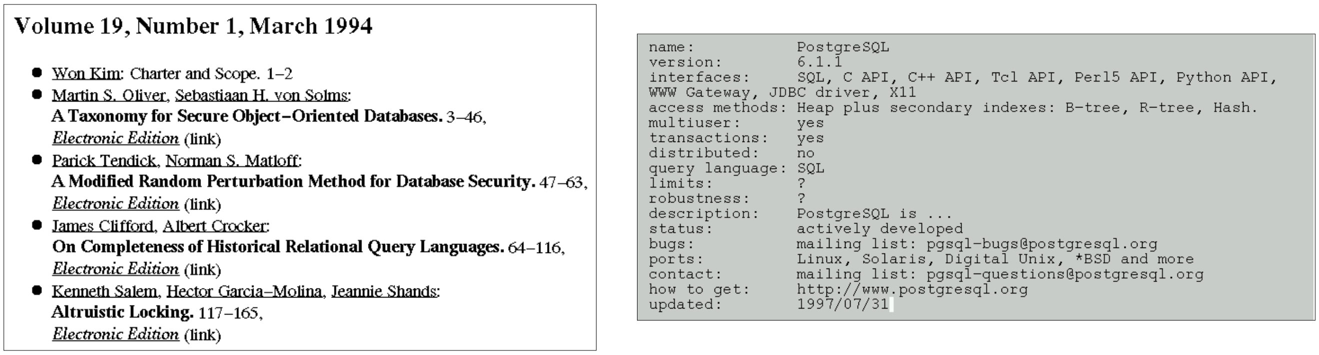
The main motivation behind data extraction tools is to ease the task of wrapper development, which is traditionally accomplished by code writing using general purpose languages such as Perl or Java.

Thus, to help the user develop wrappers for Web data, some tools present a graphical user interface (GUI) aiming at making this task easier. HTML-aware tools, NLP-based tools, wrapper induction tools, and modeling-based tools usually present a GUI. On the other hand, languages for wrapper development require the user to execute all the process manually. In the BYU tool, the ontology creation process must also be done manually by the user.

All NLP-based tools as well as the wrapper induction ones feature a GUI for the user to specify examples. In general, they allow the user to select pieces of data and to label these pieces of data properly to compose the examples.

In the case of the modeling-based tools, their GUI constitute a crucial component in the whole extraction process. In NoDoSE, the user interacts with the GUI to select and decompose regions of interest in a page and also to associate with each region a proper structure (e.g,. tuples, lists, etc.). The GUI also allows the user to test the generated wrapper against other pages and revise the previously generated extraction rule when needed. In DEByE, the user provides examples by assembling nested tables in such a way that each row of the outermost table corresponds to a distinct exam-

[2] These pages were taken respectively from the following sites: DB&LP (`http://www.informatik.uni-trier.de/~ley/db/journals/tods/`), *Free Catalog of DB Tools* (`ftp://ftp.idiom.com./pub/free-databases`), and *RISE* (`http://www.isi.edu/~muslea/RISE/`).



Volume 19, Number 1, March 1994

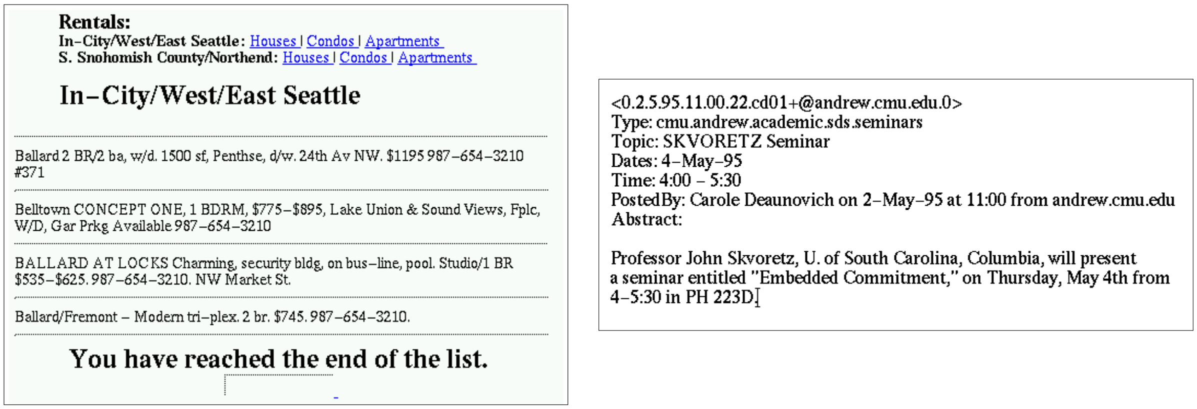
- Won Kim: Charter and Scope. 1–2
- Martin S. Oliver, Sebastiaan H. von Solms:
  A Taxonomy for Secure Object–Oriented Databases. 3–46,
  Electronic Edition (link)
- Parick Tendick, Norman S. Matloff:
  A Modified Random Perturbation Method for Database Security. 47–63,
  Electronic Edition (link)
- James Clifford, Albert Crocker:
  On Completeness of Historical Relational Query Languages. 64–116,
  Electronic Edition (link)
- Kenneth Salem, Hector Garcia–Molina, Jeannie Shands:
  Altruistic Locking. 117–165,
  Electronic Edition (link)

(a)

```
name:            PostgreSQL
version:         6.1.1
interfaces:      SQL, C API, C++ API, Tcl API, Perl5 API, Python API,
WWW Gateway, JDBC driver, X11
access methods: Heap plus secondary indexes: B-tree, R-tree, Hash.
multiuser:       yes
transactions:    yes
distributed:     no
query language: SQL
limits:          ?
robustness:      ?
description:     PostgreSQL is ...
status:          actively developed
bugs:            mailing list: pgsql-bugs@postgresql.org
ports:           Linux, Solaris, Digital Unix, *BSD and more
contact:         mailing list: pgsql-questions@postgresql.org
how to get:      http://www.postgresql.org
updated:         1997/07/31
```

(b)

Figure 1: Pages Containing Semistructured Data



Rentals:
In–City/West/East Seattle: Houses | Condos | Apartments
S. Snohomish County/Northend: Houses | Condos | Apartments

In–City/West/East Seattle

Ballard 2 BR/2 ba, w/d. 1500 sf, Penthse, d/w. 24th Av NW. $1195 987–654–3210 #371

Belltown CONCEPT ONE, 1 BDRM, $775–$895, Lake Union & Sound Views, Fplc, W/D, Gar Prkg Available 987–654–3210

BALLARD AT LOCKS Charming, security bldg, on bus–line, pool. Studio/1 BR $535–$625. 987–654–3210. NW Market St.

Ballard/Fremont – Modern tri–plex. 2 br. $745. 987–654–3210.

You have reached the end of the list.

(a)

<0.2.5.95.11.00.22.cd01+@andrew.cmu.edu.0>
Type: cmu.andrew.academic.sds.seminars
Topic: SKVORETZ Seminar
Dates: 4–May–95
Time: 4:00 – 5:30
PostedBy: Carole Deaunovich on 2–May–95 at 11:00 from andrew.cmu.edu
Abstract:

Professor John Skvoretz, U. of South Carolina, Columbia, will present
a seminar entitled "Embedded Commitment," on Thursday, May 4th from
4–5:30 in PH 223D

(b)

Figure 2: Pages Containing Semistructured Text

ple. For this, the GUI provides several operations to build nested tables (e.g., column insertion and deletion, nesting and unnesting, etc.). The GUI also features an extraction feedback mechanism that allows users to select objects imperfectly extracted and build new examples from them, thus improving the extraction performance.

W4F offers some "wizards" to assist the user in the task of wrapper generation. For helping in the writing of extraction rules for a target page, the user can select pieces of data of interest and the *extraction wizard* returns a corresponding extraction rule in HEL. This rule can them be edited and modified to cover pieces of data similar to the ones initially selected. In the case of XWRAP, the whole extraction process is guided by a GUI. It leads the user through a number of steps, implicitly selecting in each step a proper component of the library. At the end, XWRAP outputs a wrapper (coded in Java) for a specific source.

A graphical tool (ONTOS) is also provided by the BYU tool for helping the user in the process of editing an ontology.

The extraction process as performed by RoadRunner is fully automated, since it does not require any user intervention, besides selecting sample pages. Thus, from this point of view, it can be considered easier for wrapper developers than tools that require user intervention.

4.5 XML Output

XML [6] is becoming the most important standard for data representation and exchange on the Web. Due to this fact, we consider an important feature of a data extraction tool whether it provides output in XML. In this section, we describe how some of the analyzed tools provide output in XML.

In Minerva, the user has to explicitly write code to generate an output in XML. To perform this task, the user must refine the format of the extracted objects with appropriate language statements. In W4F, there is a "mapping wizard" that helps the user to create mapping rules to output the extract data in XML. XWRAP and DEByE natively provide output in XML. NoDoSE supports a variety of formats to output the data extracted from a document, among them XML and OEM.

As for the other tools studied, at the best of our knowledge, none of them provides XML output.

4.6 Support for Non-HTML Sources

A vast quantity of semistructured data stored in electronic form is not present in HTML pages, but in text files, such as e-mail messages, program code and documentation, configuration files, system logs, etc. Therefore it is very important that the data extraction tools might be able to handle such data sources.

The NLP-based tools and the BYU tool are specially suitable for non-HTML sources, since they do not depend on any kind of markup to work.

The wrapper induction tools and the modeling-based tools can also be used to extract data from non-HTML sources. These tools do not rely uniquely on HTML tags, so they are able to perform data extraction from other kinds of documents presenting some form of markup. The same can be said about Minerva and TSIMMIS, where a skilled user can code extraction rules based on any existing markup. On the other hand, as Web-OQL, W4F, XWRAP, and RoadRunner rely on the HTML tagging structure of the target page, they cannot deal with non-HTML data sources.

It is interesting to notice that tools that do not rely uniquely on HTML tags can be easily deployed for extracting data from documents that use explicit markup formats such as XML. This is useful, since it allows a uniform treatment of less structured Web sources (e.g., HTML pages) and more structured Web sources (e.g., XML documents) within a single framework.

Another interesting advantage of tools capable of dealing with non-HTML sources is that they can be used to extract from files generated by popular tools, such as Excel, in some standard export format (e.g., comma delimited). Additionally, this also opens the opportunity for interfacing with standard data access API such as ODBC and JDBC, that deal with such formats natively.

4.7 Resilience and Adaptiveness

As the structural and presentation features of Web pages are prone to frequent changes, a most needed property of wrappers is *resilience*, i.e., the capacity of continuing to work properly in the occurrence of changes in the pages to which they are targeted. It is also desirable that a wrapper built for pages of a specific Web source on a given application domain could work properly with pages from another source in the same application domain. Such a property is called *adaptiveness*.

From all of the tools covered in this paper, only the BYU tool features such properties. If an ontology for a given application domain can be constructed for capturing enough page-independent features of the data to be extracted, the wrappers generated by the tool are inherently resilient and adaptive.

In [15], the authors discuss how example-based data extraction tools such as DEByE can be extended to provide adaptiveness and, as a consequence, resilience. The general idea is, given a repository R containing data extracted by a preexisting wrapper W generated for a source S, automatically finding in R objects that can be used as examples for generating a new wrapper W' for another source S' in the same domain of S. This same general strategy could be used, in theory, with other tools such as WIEN, SoftMealy, and STALKER, which are also based on examples given by the user.

5. CONCLUSIONS

In this paper we presented a short survey of existing tools for the generation of wrappers to extract data from Web sources. In all of these tools, the main goal is to ease the task of wrapper development, which is traditionally accomplished by code writing using languages such as Perl and Java. We introduce a taxonomy for classifying the studied tools according to the type of technique they deploy for generating wrappers. However, there are cases where this simple criterion is not enough to characterize a tool. For instance, the two tools that we classified as being modeling-based, NoDoSE and DEByE, in fact use induction techniques close to the ones used by the wrapper induction tools.

We also analyze qualitatively the tools studied, by examining how they support some features that we regard as most important to accomplish the generation of wrappers and the data extraction process performed by them. Table 1 presents a summary of this analysis. In the column Ease of Use, the number of "+" expresses the easiness of using the corresponding tool. In the last column of this table, SD stands for "Semistructured Data" and ST stands for "Semistructured Text". Note that in this table we omit information on resilience and adaptiveness, since among all tools studied only the BYU tool provides this feature natively.

In Figure 3 we provide a distinct perspective of the summary in Table 1. In this figure, we dispose the groups of data extraction tools identified in Section 2 into a bi-dimensional space where the horizontal axis represents the degree of automation of the tools belonging to a group and the vertical axis represents the degree of flexibility of the tools.

Observing Figure 3, it can be seen that, at least in the present state of the art of Web data extraction, there is an inherent trade-off between the degree of automation and the degree of flexibility of current data extraction tools presented in the literature. This is explained by the fact that tools presenting a higher degree of automation implement heuristics based on hypotheses (sometimes too strong ones) about the features of the target pages (e.g., type of content, formatting, etc.). This alleviates users from providing additional information on these features, but requires the tuning of these tools for dealing with them, compromising their generality. On the other hand, relying on the user to provide specific information on the features of the data to be extracted results in tools that require less predefined heuristics to be used, so leading to flexibility.

To exemplify such a trade-off, consider that HTML-aware tools use a number of heuristics to infer a plausible schema from the HTML tag hierarchy of a target page. Thus, some hypotheses on the use of HTML constructs to structure data must be assumed. On the contrary, modeling-based tools require the user to provide a schema for the data to be extracted according to his own perception of how the data is organized. This reduces the dependency of such tools on the HTML features of the target pages.

The classification and qualitative analysis presented in this paper is part of a more comprehensive work we are carrying out for comparing existing approaches and tools for Web data extraction. In particular, of great interest is a quantitative analysis of experimental results obtained with some of the tools here mentioned, which is the main emphasis of our current work and whose preliminary results are reported in [33].

Tools		Degree of Automation	Support for Complex Objects	Ease of Use	XML Output	Support for Non-HTML Sources	Type of Page Contents
Languages	Minerva	Manual	Coding	+	Yes	Partial	SD
	TSIMMIS	Manual	Coding	+	No	Partial	SD
	Web-OQL	Manual	Coding	+	No	None	SD
HTML-aware	W4F	Semi-Automatic	Coding	++	Yes	None	SD
	XWRAP	Automatic	Yes	++++	Yes	None	SD
	RoadRunner	Automatic	Yes	++++	No	None	SD
NLP-based	WHISK	Semi-Automatic	No	++	No	Full	ST
	RAPIER	Semi-Automatic	No	++	No	Full	ST
	SRV	Semi-Automatic	No	++	No	Full	ST
Induction	WIEN	Semi-Automatic	No	++	No	Partial	SD
	SoftMealy	Semi-Automatic	Partial	++	No	Partial	SD
	STALKER	Semi-Automatic	Yes	++	No	Partial	SD
Modeling-based	NoDoSE	Semi-Automatic	Yes	+++	Yes	Partial	SD
	DEByE	Semi-Automatic	Yes	+++	Yes	Partial	SD
Ontology-based	BYU	Manual	Coding	++	No	Full	ST/SD

Table 1: Summary of the Qualitative Analysis

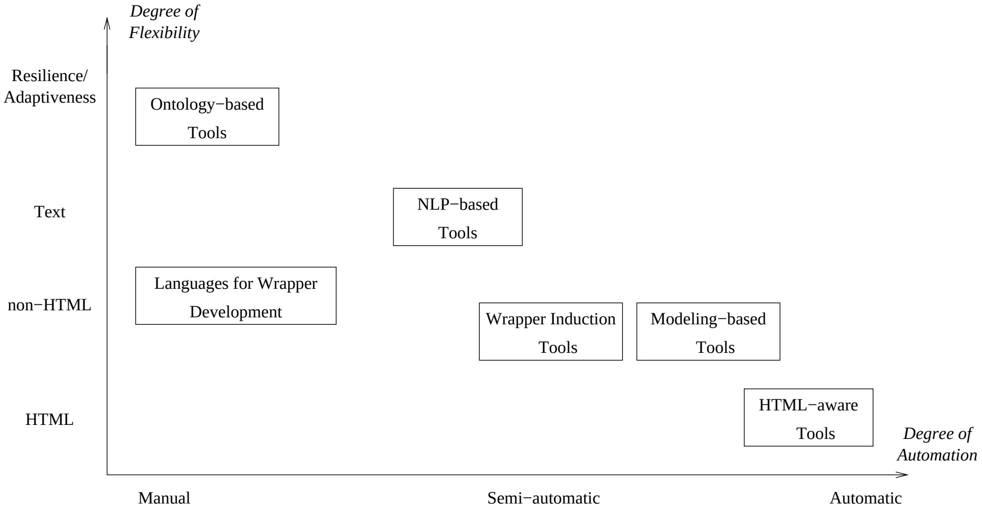


Figure 3: Graphical Perspective of the Qualitative Analysis

6. REFERENCES

[1] Abascal, R., and Sánchez, J. A. X-tract: Structure extraction from botanical textual descriptions. In *Proceeding of the String Processing & Information Retrieval Symposium and International Workshop on Groupware, SPIRE/CRIWG* (Cancún, Mexico, 1999), pp. 2–7.

[2] Abiteboul, S. Querying semi-structured data. In *Database Theory - ICDT'97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings* (1997), F. N. Afrati and P. Kolaitis, Eds., vol. 1186 of *Lecture Notes in Computer Science*, Springer, pp. 1–18.

[3] Adelberg, B. NoDoSE – A tool for semi-automatically extracting structured and semistructured data from text documents. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Seattle, WA, 1998), pp. 283–294.

[4] Arocena, G. O., and Mendelzon, A. O. WebOQL: Restructuring documents, databases, and webs. In *Proceedings of the 14th International Conference on Data Engineering* (Orlando, FL, 1998), pp. 24–33.

[5] Baumgartner, R., Flesca, S., and Gottlob, G. Visual Web information extraction with Lixto. In *Proceedings of the 26th International Conference on Very Large Data Bases* (Rome, Italy, 2001), pp. 119–128.

[6] Bray, T., Paoli, J., and Sperberg-McQueen, M. Extensible markup language (XML) 1.0. http://www.w3.org/TR/REC-xml.

[7] Brin, S., Motwani, R., Page, L., and Winograd, T. What can you do with a Web in your pocket? *Data Engineering Bulletin 21*, 2 (1998), 37–47.

[8] Califf, M. E., and Mooney, R. J. Relational Learning of Pattern-Match Rules for Information Extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence* (Orlando, FL, 1999), pp. 328–334.

[9] Crescenzi, V., and Mecca, G. Grammars have

exceptions. *Information Systems 23*, 8 (1998), 539–565.

[10] Crescenzi, V., Mecca, G., and Merialdo, P. RoadRunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 26th International Conference on Very Large Data Bases* (Rome, Italy, 2001), pp. 109–118.

[11] Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., kai Ng, Y., Quass, D., and Smith, R. D. Conceptual-model-based data extraction from multiple-record Web pages. *Data and Knowledge Engineering 31*, 3 (1999), 227–251.

[12] Embley, D. W., Jiang, Y. S., and Ng, Y.-K. Record-boundary discovery in Web documents. In *Proceedings ACM SIGMOD International Conference of Management of Data* (Philadelphia, PA, 1999), pp. 467–478.

[13] Florescu, D., Levy, A. Y., and Mendelzon, A. O. Database techniques for the World-Wide Web: A survey. *SIGMOD Record 27*, 3 (1998), 59–74.

[14] Freitag, D. Machine Learning for Information Extraction in Informal Domains. *Machine Learning 39*, 2/3 (2000), 169–202.

[15] Golgher, P. B., da Silva, A. S., Laender, A. H. F., and Ribeiro-Neto, B. A. Bootstrapping for Example-Based Data Extraction. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management* (Atlanta, GA, 2001), pp. 371–378.

[16] Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breunig, M., and Vassalos, V. Template-based wrappers in the TSIMMIS system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Tucson, AZ, 1997), pp. 532–535.

[17] Hammer, J., McHugh, J., and Garcia-Molina, H. Semistructured data: The TSIMMIS experience. In *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems* (St. Petersburg, Russia, 1997), pp. 1–8.

[18] Hsu, C.-N., and Dung, M.-T. Generating finite-state transducers for semi-structured data extraction from the Web. *Information Systems 23*, 8 (1998), 521–538.

[19] Huck, G., Fankhauser, P., Aberer, K., and Neuhold, E. J. Jedi: Extracting and synthesizing information from the Web. In *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems* (New York City, NY, 1998), pp. 32–43.

[20] Kushmerick, N. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence Journal 118*, 1-2 (2000), 15–68.

[21] Laender, A. H. F., Ribeiro-Neto, B., and da Silva., A. S. DEByE – Data Extraction By Example. *Data and Knowledge Engineering 40*, 2 (2002), 121–154.

[22] Laender, A. H. F., Ribeiro-Neto, B., da Silva, A. S., and Silva, E. S. Representing Web Data as Complex Objects. In *Electronic Commerce and Web Technologies*, K. Bauknecht, S. K. Mandria, and G. Pernul, Eds. Springer, Berlin, 2000, pp. 216–228.

[23] Liu, L., Pu, C., and Han, W. XWRAP: An XML-enabled wrapper construction system for Web information sources. In *Proceedings of the 16th International Conference on Data Engineering* (San Diego, CA, 2000), pp. 611–621.

[24] Ludäscher, B., Himmeröder, R., Lausen, G., May, W., and Schlepphorst, C. Managing semistructured data with FLORID: A deductive object-oriented perspective. *Information Systems 23*, 8 (1998), 589–613.

[25] Mecca, G., Atzeni, P., Masci, A., Merialdo, P., and Sindoni, G. The Araneus Web-Base Management System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Seattle, WA, 1998), pp. 544–546.

[26] Muslea, I. RISE: Repository of online information sources used in information extraction tasks. *http://www.isi.edu/ muslea/RISE/.*

[27] Muslea, I. Extraction Patterns for Information Extraction Tasks: A Survey. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction* (Orlando, FL, 1999), pp. 1–6.

[28] Muslea, I., Minton, S., and Knoblock, C. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems 4*, 1/2 (2001), 93–114.

[29] Papakonstantinou, Y., Garcia-Molina, H., and Widom, J. Object Exchange Across Heterogenous Information Sources. In *Proceedings of 11th International Conference on Data Engineering* (Taipei, Taiwan, 1995), pp. 251–260.

[30] Ribeiro-Neto, B., Laender, A. H. F., and da Silva, A. S. Extracting semi-structured data through examples. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management* (Kansas City, MO, 1999), pp. 94–101.

[31] Sahuguet, A., and Azavant, F. Building intelligent Web applications using lightweight wrappers. *Data and Knowledge Engineering 36*, 3 (2001), 283–316.

[32] Soderland, S. Learning information extraction rules for semi-structured and free text. *Machine Learning 34*, 1–3 (1999), 233–272.

[33] Teixeira, J. S. A Comparative Study of Approaches for Semistructured Data Extraction. Master's thesis, Department of Computer Science, Federal University of Minas Gerais, Brazil, 2001. In Portuguese.

[34] World Wide Web Consortium. W3C. The Document Object Model. *http://www.w3.org/DOM.*