

Extracting Semi-Structured Data Through Examples*

Berthier Ribeiro-Neto, Alberto H.F. Laender, Altigran S. da Silva[†]

Department of Computer Science
Federal University of Minas Gerais
31270-901 Belo Horizonte MG Brazil
{berthier,laender,alti}@dcc.ufmg.br

Abstract

In this paper, we describe an innovative approach to extracting semi-structured data from Web sources. The idea is to collect a couple of example objects from the user and to use this information to extract new objects from new pages or texts. To perform the extraction of new objects, we introduce a bottom-up extraction strategy and, through experimentation, demonstrate that it works quite effectively with distinct Web sources, even if only a few examples are provided by the user.

1 Introduction

As the data on the Web grows at explosive rates, a tremendous research effort has been initiated to make such data available, usually in some structured form such as a table, for querying and further manipulation. The main motivation is that structured data allow asking queries which cannot be asked with data in text form. For instance, for a hotel information site, a typical example of such a query is: *retrieve all hotels in downtown New York with daily rates smaller than US\$ 120.00.*

In this work, we describe an innovative example-based approach to extract data from a set of Web pages for populating a database. In this approach, we assume that the Web pages present some inherent structure which can be readily recognized. Since this structure appears implicitly on the pages and might vary from one page to another, we say that the data is *semi-structured* [9].

To extract the data, we need some description of what to extract. A common approach to providing such description

is to build a specific grammar which details the surroundings of each piece of data to extract. In this work, however, we consider a new approach in which the description of what to extract is fully based on a small set of examples provided by the user. We demonstrate that just a couple of examples are sufficient for extracting hundreds of objects from new Web pages. Our approach is remarkably distinct from previous works because it relies on the user's perception of the structure of the objects. That is, instead of trying to derive structural (semantic) information from the text format (syntax), we induce the user to inform the structure as she/he perceives it. Furthermore, our approach is not tied to any specific formatting system (e.g. HTML, XML, LaTeX, etc.). Instead, it takes advantage of any markups surrounding the data of interest.

To perform the extraction of data, we introduce a bottom-up strategy that, given a set of Web pages as input, recognizes objects matching the given examples and extracts them. This strategy, as we demonstrate by experimentation, works very well when dealing with complex objects which present a non-flat structure, even if the structure of the objects presents variations in comparison to the structures of the objects given as examples. Our results show that this bottom-up example-based extraction strategy is highly effective with various types of Web sources.

The paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe the data extraction by example approach. In Section 4, we introduce the notion of *Object Extraction Patterns*, which is a central concept in our approach. Section 5 presents our example-based bottom-up extraction strategy. Section 6 discusses our experimental results. Finally, Section 7 presents our conclusions.

2 Related Work

A number of different approaches has been proposed to extracting data from Web sources [5, 6, 7, 14, 16, 18, 21], as we now discuss. The most common of such approaches involves the use of wrappers [6, 7, 15, 16]. A wrapper parses pages from specific Web sources based on some kind of grammar and maps these data into a pre-specified format. Originally, wrappers were manually written for a specific source [7, 11] but, due to the obvious limitations of this approach, efforts have been made to automatize the wrapper generation process [6, 15, 16]. Although wrappers provide an effective approach to data extraction from Web sources, they have two

*This work is partially funded by MCT/FINEP/PRONEX, under Project SIAM (grant 76.97.1016.00), and by individual research grants from CNPq and CAPES.

[†]On leave from University of Amazonas, Brazil.

major drawbacks. Firstly, they require a previous knowledge of the structure of the data source. Secondly, they are very rigid and additional work might be required to adapt the wrapper when the source changes.

An alternative approach to extracting data from Web sources is based on natural language processing (NLP). In such an approach, NLP techniques are used to find relevant fragments that can be extracted from a source document [12]. The system described in [21] is an example of a tool that uses such techniques. An NLP-based approach is effective but is very specific and usually demands a large number of training examples.

An ontology-based approach to extracting data from Web sources is presented in [13, 14]. This approach uses a semantic data model to provide an ontology that describes the data of interest and its location on the source pages, including relationships, lexical appearances, and context keywords. By parsing this ontology, a relational database schema and a constant/keyword recognizer are automatically generated, which are then used to extract the data that will populate the database. Although this approach can be quite effective, its major drawback is that it requires the ontology to be manually constructed for a given domain.

An approach more related to ours is the one adopted by the NoDoSe tool [5]. This tool provides a graphical interface which the user uses to decompose a given document (e.g., a Web page) into a hierarchy that describes its structure. Additional documents of the same type are then provided to the tool and automatically parsed. Any errors are corrected by inspecting the results (using the interface), modifying the hierarchy that describes the documents, and reparsing them. The process is completed when all of the documents have been successfully parsed. This approach is effective for a large class of textual documents but, unlike ours, the generation of the extraction patterns is semi-automatic. Moreover, any mismatches found on these patterns must be corrected during the extraction process which means that the entire process is user-assisted.

Finally, a more specific but very interesting approach to extracting tabular data from Web sources is provided by TINTIN (Table INformation-based Text INquiry) [18]. This tool extracts tabular data from unstructured documents based on a purely structural analysis of such documents. Heuristics are used to recognize the headings and the values that compose the different columns of a table.

3 Data Extraction by Example

Consider a portion of a Web page obtained from the DB&LP site [3] as illustrated in Figure 1. We notice that there is an inherent structure to the text on the page. For instance, we are able to distinguish four *objects* and, for each object, we identify attributes such as a list of author's names and title. Such structure has not been declared anywhere but is clearly identifiable. Texts or pages which present such type of inherent structure are said to be *data rich* and *narrow in ontological breadth* [14]. Such pages constitute the target of our approach and are referred to simply as data rich pages.

Given a set of data rich pages, we investigate how to extract (from them) objects and their attributes such that they can, for example, be inserted into (nested) tables for later querying. If properly done, this would allow retrieving information which cannot be obtained with standard text searching techniques. For instance, in Figure 1, one might

Volume 20, Number 3, September 1995

- Weidong Chen:
Query Evaluation in Deductive Databases with Alternating Fixpoint Semantics. 239–297,
[Electronic Edition \(link\)](#)
- Yannis E. Ioannidis, Raghu Ramakrishnan:
Containment of Conjunctive Queries: Beyond Relations as Sets. 288–324,
[Electronic Edition \(link\)](#)
- Dennis Shasha, François Liribat, Eric Simon, Patrick Valduriez:
Transaction Chopping: Algorithms and Performance Studies. 325–363,
[Electronic Edition \(link\)](#)
- Stefano Ceri, Piero Fraternali, Stefano Paraboschi, Letizia Tanca:
Addendum to 'Automatic Generation of Production Rules for Integrity Maintenance'.
364,
[Electronic Edition \(link\)](#), see *TODS* 19(3): 367–422 (1994)

Figure 1: Web page from the DB&LP site illustrating a data rich example (ACM TODS).

be interested in all the titles which have been published by a given author alone after March 1997. Notice that this information is present on the page but cannot be obtained using standard text retrieval techniques.

To be able to extract information from a set of data rich pages, we need some type of description of what to extract. For instance, we could assume the existence of a grammar detailing how to parse and recognize tokens for insertion on a table. In the example of Figure 1, such grammar could specify that names of authors appear after a black dot and are separated by commas. Further, the grammar could state that the title appears in the line immediately after the line containing names of authors. However, as already mentioned, the main grammar-based approaches are too rigid for processing typical text which appear in practical situations (particularly, in the Web). For instance, an entry might be missing the black dot which identifies the object, might be missing information on authors, or might misplace the information on the title such that it appears prior to the information on the authors. To deal with such situations, the designer of the grammar would have to anticipate which exceptions could occur in practice and adapt the grammar accordingly. Clearly, such task might be quite hard in loose domains such as the Web.

In this work, we take a different approach and assume that a well informed user simply specifies examples of objects to extract. Using these examples, we devise a strategy to extracting the data from new pages (with similar structure) in the presence of mismatches, variation on the ordering of attributes, and an inconsistent (implicit) structure. The user provides such examples through a graphical interface by cutting pieces of data from the page and assembling an example object. We expect that a couple of example objects should suffice to allow processing hundreds of new (but similar in structure) pages.

We consider that the example object the user provides is a complex object with a hierarchical structure. For instance, for the Web page in Figure 1, the user could specify a 2-level hierarchical example as illustrated in Figure 2. In this case, the user specified a single example of a paper with 4 authors, 39 pages, and which appeared in September of 1995. Once an object is properly structured, it can be directly inserted into a nested table for later querying, as illustrated in Figure 3. Further, this nested table can be flattened for querying as a standard relational table. For each piece of data in the example object in Figure 2, we assume that we know the position in the original page where it came from. For instance, the author name 'Dennis Shasha' initiates at the second non-blank character of the ninth non-blank line in Figure 1. Considering that the user provides the examples through a graphical interface that allows him/her to mark

Edition															
Volume	Number	Date	Article												
20	3	September 1995	<table border="1"> <thead> <tr> <th>Author</th> <th>Title</th> <th>Pages</th> </tr> </thead> <tbody> <tr> <td>{Denis Shasha, . . ., Patrick Valduriez}</td> <td>Transaction . . .</td> <td>325-363</td> </tr> <tr> <td>{Weidong Chen}</td> <td>Query Evaluation. . .</td> <td>239-297</td> </tr> <tr> <td>. . .</td> <td>. . .</td> <td>. . .</td> </tr> </tbody> </table>	Author	Title	Pages	{Denis Shasha, . . ., Patrick Valduriez}	Transaction . . .	325-363	{Weidong Chen}	Query Evaluation. . .	239-297
			Author	Title	Pages										
			{Denis Shasha, . . ., Patrick Valduriez}	Transaction . . .	325-363										
{Weidong Chen}	Query Evaluation. . .	239-297													
.													
20	4	December 1995	<table border="1"> <thead> <tr> <th>Author</th> <th>Title</th> <th>Pages</th> </tr> </thead> <tbody> <tr> <td>{IMin A. Chen, . . ., Denis McLeod}</td> <td>An Execution Model . .</td> <td>365-413</td> </tr> <tr> <td>{Piero Fraternali, Letizia Tanca}</td> <td>A Structured. . .</td> <td>414-471</td> </tr> <tr> <td>. . .</td> <td>. . .</td> <td>. . .</td> </tr> </tbody> </table>	Author	Title	Pages	{IMin A. Chen, . . ., Denis McLeod}	An Execution Model . .	365-413	{Piero Fraternali, Letizia Tanca}	A Structured. . .	414-471
			Author	Title	Pages										
			{IMin A. Chen, . . ., Denis McLeod}	An Execution Model . .	365-413										
{Piero Fraternali, Letizia Tanca}	A Structured. . .	414-471													
.													
.												

Figure 3: Nested table containing extracted objects

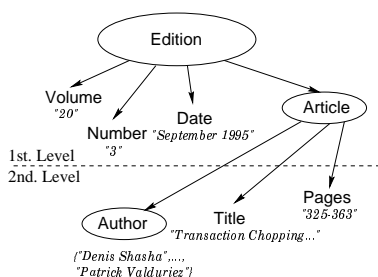


Figure 2: Object provided as example for the Web page in Figure 1.

pieces of data on a page and drag them, this positional information is trivially generated by storing the positions (in the text) of the pieces marked by the user. Positional information is easy to obtain but represents a crucial piece of evidence for assisting with the data extraction process as we later demonstrate.

Given a very small set of example objects, there are several possible strategies for extracting data from new pages with similar structure. A simple strategy is the one roughly described by the algorithm below (consider for a moment that the examples provided by the user are flat, i.e., they have a single level hierarchical structure).

```

begin
  Let  $G$  be a set of pages given as input;
  Let  $O_e$  be an example object given as input;
  foreach attribute  $A$  of  $O_e$  do
    Determine a local text context for the piece of data
    associated with  $A$ ;
  end
  Combine all the local contexts and generate
  a context for the object  $O_e$ ;
  Use the context description for  $O_e$  to recognize and
  extract new objects in pages of  $G$ ;
end

```

The algorithm works by assembling a context for an object and using this context description to identify new objects in new pages. The context describes the surroundings of the object provided as example by the user. Despite its simplicity, this extraction strategy, which we call *top-down* [19], works well with pages that are well structured (i.e., that are data rich and present none or little variation in their structure). With pages with variable structure (which are quite common in the Web), a distinct *bottom-*

up extraction strategy is more appropriate, as discussed in Section 5.

Independently of the extraction strategy used to effectively extract the objects, our example-based approach requires an environment which allows the specification of examples and the extraction of the semi-structured data. We devised such an environment which we call *DEByE (Data Extraction by Example)**. The major components for this environment are presented in Figure 4. Basically, the DE-

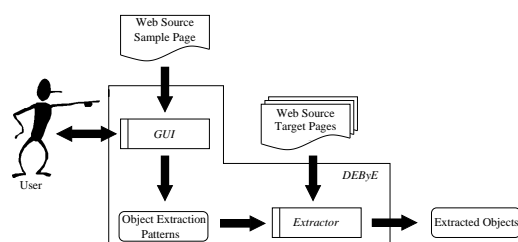


Figure 4: Components of the DEByE environment and their role in a data extraction process.

ByE environment consists of a *Graphical User Interface (GUI)* module and an *Extractor* module. These two modules communicate through an intermediary data structure called *Object Extraction Patterns* that we discuss in more detail in Section 4.

The *GUI* module provides the user with a Java interface that he uses to assemble a couple of example objects. The assembled objects are then used to generate patterns for extracting new objects. A sample screen of the DEByE *GUI* is presented in Figure 5. The sample page appearing in the “Source Window” is from the Murder by the Book site, which is a Web bookstore. In this case, the user has provided as examples a title by “Agatha Christie” and two titles by “Leslie Charteris”. These examples are specified by cutting data from the “Source Window” and pasting these data in the “Table Window”. The structure of the table shown in “Table Window” is defined by the user using operations provided by the GUI. The assembled objects are then used to generate patterns for extracting new objects.

The *Extractor* module takes the generated patterns and applies them to new pages from the target Web source. It then proceeds to identify new objects which are then extracted and provided as output. The *Extracted Objects* are

*This name is a homage to Moshé Zloof, creator of the QBE language [22], who suggested the paradigm we use to specify the data to be extracted from Web sources.

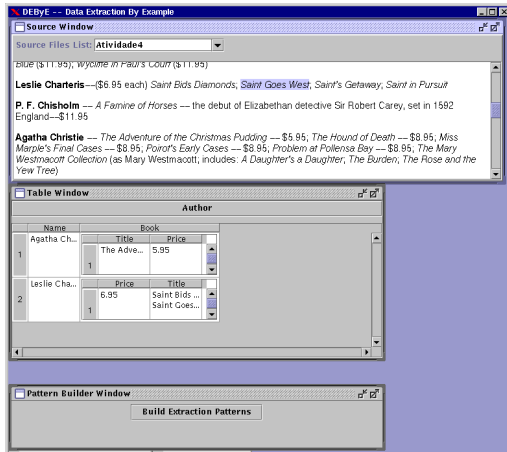


Figure 5: A sample screen of the DEByE GUI.

stored in regular text files using a XML-based format that allows for easy conversion to other formats or for insertion in (nested) tables for later querying.

In this paper, our main focus is on describing a particular extraction strategy, called *bottom-up* strategy, which is used by the *Extractor* module. This strategy is discussed in detail in Section 5. For details on the DEByE GUI, its features, and implementation, we refer the reader to [20]. Before discussing the bottom-up extraction strategy, we discuss in next section how the example objects specified by the user are used to generate *Object Extraction Patterns*.

4 Object Extraction Patterns

Object Extraction (OE) patterns are patterns used by the DEByE *Extractor* module to recognize and extract new objects from new Web pages. Each OE pattern describes the hierarchical structure of an example object specified by the user. The hierarchy associated with an OE pattern is composed of nodes structured in levels as shown in Figure 2. The nodes at the bottom of the hierarchy are used to identify the atomic objects which compose a more complex object and are called *AVP Patterns*. In this section, we first define AVP patterns and then describe the OE patterns in more detail.

4.1 AVP Patterns

To specify an example, the user selects pieces of data which he uses as atomic components of an example object. Each piece of data selected is called a *value* while each atomic component is referred to as an *attribute*. Thus, we use the terminology *attribute-value pair* (or, AVP) to refer to an attribute and one of its values.

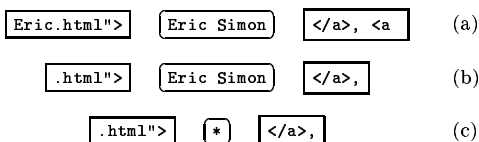


Figure 6: Two possible local contexts and a pattern for the AVP value *Eric Simon*.

The recognition and extraction of (implicit) objects pre-

sent on a page are based on the notion of a local context for each attribute-value pair (AVP). Local contexts are derived from the text in which the AVP occurs as follows. Consider the position in the text (or Web page) of an AVP value selected by the user. The text portions surrounding the AVP value constitute a passage (or window) [10, 17] which can be used as a local context. For instance, Figure 6 illustrates the AVP value *Eric Simon*, as it occurs in the HTML source of the page shown in Figure 1, along with passages which can be used as its local context. Using this context information we build a *pattern* (also called *AVP pattern*) which can be later used to identify authors names such as *Eric Simon*. Figure 6c illustrates an AVP pattern corresponding to the AVP value *Eric Simon* using the local context shown in Figure 6b. The symbol “*” matches a sequence of characters of any length. Note that if we had used the local context of Figure 6a, the resulting AVP pattern would include context information that is too specific and, therefore, it would not retrieve any author name other than *Eric Simon*. Thus, to be able to generate AVP patterns for extracting other author names, we must adopt a more flexible pattern generation strategy. This can be accomplished as follows.

Given an AVP selected by the user, we determine a passage surrounding this AVP value in the text. In this work, we adopt symmetric passages composed of W text tokens to the right and W text tokens to the left of the AVP value. Due to the application of some simple heuristics for token identification (for instance, a run of spaces is considered a single token), an AVP pattern might result asymmetric.

The width W of an AVP pattern is determined automatically for each type of AVP. We start with a small pattern (which is usually too general) and make it more specific by increasing its width W , until we find a pattern that provides a good fit.

The resulting AVP patterns are then used to compose an *Object Extraction (OE) pattern*.

4.2 OE Patterns

As we have already said, OE patterns are the patterns used by the extractor module to find and extract new objects from new Web pages. They include information on the structure of the objects and on the AVP patterns associated with the attributes of these objects. For pages with non-homogeneous objects (as the pages for the Murder by the Book site), the user might have to specify more than one example object for allowing extraction of a larger fraction of the (implicit) objects in new pages. For instance, in Figure 5 the user specified a second example object to indicate that, in the case of the author *Leslie Charteris*, the price of all her books is the same.

For each of the example objects in Figure 5, a distinct OE pattern is generated. Figure 7 illustrates the two OE patterns corresponding to the two example objects in Figure 5. The *Extractor* module illustrated in Figure 4 receives OE patterns from the GUI module and matches them against new incoming Web pages.

5 Extraction Strategy

In this section, we discuss a *bottom-up* strategy that, given an OE pattern, extracts from a Web page a set of objects matching it. Before discussing this strategy, we define some

```

<OBJECTS>
  <TUPLE type="Author">
    <ATOM type="name">
      <PATTERN><![CDATA[<b>[a]*?([X]+)?[a]*?(?=</b>)]]></PATTERN>
    </ATOM>
    <LIST type="Book">
      <TUPLE type="Book">
        <LIST type="Title">
          <ATOM type="Title">
            <PATTERN>
              <![CDATA[ -- [a]*?<D>[a]*?([XF]+)?[a]*?(?=</I>[a]*? -- \$)]]>
            </PATTERN>
          </ATOM>
        </LIST>
        <ATOM type="Price">
          <PATTERN><![CDATA[--\$[a]*?([0-9]+\.[0-9]+)[a]*?(?=<P>)]]></PATTERN>
        </ATOM>
      </TUPLE>
    </LIST>
  </TUPLE>
</OBJECTS>

```

Figure 7: Object extraction pattern generated for the example objects illustrated in Figure 5.

basic concepts and terminology required to deal with the structure of complex objects.

5.1 Basic Concepts and Terminology

Since the structure of Web page objects is not always flat, it is necessary to introduce the concept of a complex object with a hierarchical structure as we now do. Instead of formally defining a complex object, we introduce the terminology through the discussion of an example. The example we adopt is the complex object labeled Edition in Figure 2.

The object Edition in Figure 2 is composed of four other objects which are called component objects or simply *components*. The first three component objects (i.e., Volume, Number, and Date) are atomic and are referred to as attribute (or atom) objects or simply *attributes* (or *atoms*). The fourth component (i.e., Article) is a list object or simply a *list*. This list is formed by several *identical* complex objects. Each of these objects is composed of a list (i.e., Author) and two atoms (i.e., Title and Pages). The list Author is itself formed by atomic objects.

To simplify the discussion below, we first describe the bottom-up strategy for flat objects only. Following, we describe a general version that can be used to extract objects presenting multi-level structures.

5.2 Bottom-up Extraction Strategy

The main feature of our bottom-up extraction strategy is that it recognizes and extracts atomic object components (which lie at the bottom of the hierarchical structure of a complex object) prior to the recognition of the object itself. The component objects are then used to assemble the object through a bottom-up composition operation. The details are as shows the Algorithm in Figure 8.

Our bottom-up extraction strategy is considerably more complex than the top-down strategy outlined in Section 3. Instead of assembling a pattern for the whole object, we work with the AVP patterns composing an OE pattern O_e . For each AVP pattern, we obtain all strings matching it within the current page g and store them in AVP_BAG, along with positional information. The strings in this set are then used to compose the objects.

Since the variable AVP_BAG is a set, there is no notion of order among the strings in this set. Further, the objects being composed might result incomplete in the sense that they do not include all the components of the OE pattern

BU-Extraction

begin

Let G be a set of Web pages given as input;
Let O_e be an OE pattern given as input;

foreach page g in G **do**

/* Extracts AVP from pages and store them in AVP_BAG */
AVP_BAG \leftarrow \emptyset ;

foreach AVP attribute A in O_e **do**

Let p be the AVP pattern corresponding to A ;

foreach string s in g matching p **do**

Let l be the position of s in g ;

AVP_BAG \leftarrow AVP_BAG \cup $\{(A, s, l)\}$;

end

end

/* Assemble objects using triples from AVP_BAG. */

/* The assembled objects have a structure similar to O_e . */

Let (A, s, l) be the triple in AVP_BAG with lowest l ;

MARK \leftarrow A ;

while AVP_BAG \neq \emptyset **do**

if $A = \text{MARK}$ then

create a new object O with the structure of O_e ;

end

Assemble s as the value for attribute A into O ;

remove (A, s, l) from AVP_BAG;

Let (A, s, l) be the triple in AVP_BAG with lowest l ;

end

end

end

Figure 8: Details of the bottom-up extraction strategy

O_e . In fact, in the bottom-up strategy a new object will be recognized and composed even if (a) its components appear out of order (for instance, the title appears before the author names) and (b) it is missing some of its components. These properties of our bottom-up strategy provide for greater flexibility and naturally take into account the notion of partial object recognition.

We observe that there is no way of telling when the strings in the set AVP_BAG are related to a same object. The reasons are: (1) we have collected the AVPs without paying attention to the structure information provided by the OE pattern O_e and (2) a new object might be missing one or more of its components. To avoid mixing up components of two distinct objects, we use the first attribute found in the page as a mark, i.e., every string recognized as a value of this attribute indicates the beginning of a new object. Thus all attributes values occurring between such marks are assumed to belong to a same object.

Even being more complex, the bottom-up strategy as describe so far can only deal with flat structures. The reason is that it mixes up all the AVP and thus, is unable to distinguish more complex objects whose components are, for instance, lists. To deal with this situation, we need to generalize our strategy such that it also takes into account OE patterns presenting a more complex structure.

5.3 Generalizing the Bottom-up Extraction Strategy

To generalize the bottom-up strategy for objects presenting a non-flat structure, consider that the OE pattern O_e is in fact a sub-hierarchy of a more complex OE pattern O_f . When all objects matching O_e are assembled, they will be used to composed the more complex objects, i.e., the objects matching O_f , in the same fashion as the extracted AVPs were used to compose objects matching O_e . The same is done for all the sub-hierarchies of O_f , along with any attribute composing it. Cases of objects with more nesting levels are naturally treated because O_f can itself be a sub-

hierarchy of a more complex OE pattern and so on. We note that the component objects recognized might be approximate (in the sense that their sub-components might be out of order or missing).

The whole procedure works from the lower levels of the hierarchy to the upper levels. The frontier of a complex object can be delineated through proper markers in a similar way as used in the simple bottom-up strategy. We notice, however, that in this case complex objects can be also regarded as marks. By doing so, we are able to identify complex component objects and compose them properly, naturally recognizing partial objects without compromising its time efficiency. This is the main idea and the details are not provided here. This increased flexibility of the bottom-up strategy is quite powerful in practice, particularly with Web pages, as we demonstrate in Section 6.

The top-down strategy outlined in Section 3 recognizes objects in their entirety. Thus, recognition of partial objects (i.e., objects which are missing a component such as a date) is not done. Further, objects which contain components out of order (for instance, the title appearing before the author names) are also not recognized. To recognize partial objects, the top-down strategy depends on a potentially large set of example object patterns. To illustrate, consider a two level example object with three atomic components labeled O_a , O_b , and O_c . Retrieval of all possible partial matching objects would require 7 distinct object patterns (one pattern for the complete object, three patterns to indicate the absence of a single component, and three patterns to indicate the absence of two components). For large example objects, the number of cases might be high which makes the extraction procedure inefficient in time (because each new page has to be processed independently for each example pattern). In case the example object is described by a multi-level hierarchy, the set of example patterns becomes more complex and larger which further diminishes the overall effectiveness (in practice) of the whole extraction procedure.

Figure 9 illustrates graphically the behavior of the bottom-up extraction strategy in comparison to the top-down strategy. The bottom-up strategy assembles complex objects through a composition of simpler component object matches, while the top-down strategy recognizes entire complex objects and decomposes them into simpler components.

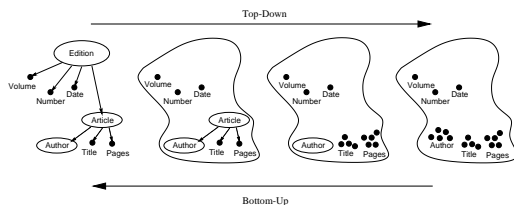


Figure 9: Behavior of the bottom-up extraction strategy.

6 Results

In this section we discuss the application of our extraction strategy to HTML pages collected from popular Web sources. The results are presented in two parts. In the first part, we use Web sources whose (implicit) objects are structured in one-level. In the second part, we use a Web source whose (implicit) objects are structured in two-level hierarchies.

For the first part of our experiments, we use three Web sources: (a) CDNow [2], (b) Travelocity [4], and (c) Amazon [1]. Figure 10 illustrates excerpts of pages from these three sources. CDNow is a site whose pages present basically the same structure and are composed of homogeneous objects. Travelocity is a site where pages present objects without a clearly defined structure. Amazon is a site whose pages include objects which might be incomplete (i.e., some components might be missing).

Placebo Without You I'm Nothing	\$16.97	\$11.88
Portishead <i>Pyg</i>	\$16.97	\$11.88
Louis Prima <i>Collectors Series</i>	\$11.97	\$8.28
Queen <i>Circus! Hits I & II</i>	\$29.97	\$20.98
R.E.M. <i>Up</i>	\$16.97	\$11.88

(a) CDNow

San Juan's Castle - Barbados \$333- (5 nts) Stay at this legendary landmark castle on one of the world's most beautiful beaches! Dive into 3 pools, soak in the whirlpool, play tennis on lighted courts, sail, snorkel, waterski, and work out in the fitness room. All on 72 landscaped acres.
3-Night Southern Caribbean Cruise \$349- Now you can explore the faraway islands of the Southern Caribbean on a three-night cruise - and save a minimum of \$140 off the brochure rate! Aboard the <i>Norvic Express</i> , you will depart San Juan and visit St. Thomas and St. Maarten.
Wynham Aruba Beach Resort & Casino \$366- (5 nts) Couples, singles and families alike will love this resort located on Aruba's most exclusive beach. Enjoy swimming, water sports, tennis and nearby championship golf. And the Kids Klub is fun for all ages.

(b) Travelocity

Internet Publishing Kit
Hardcover / Published 1995
Our Price: \$149.95 (Special Order)
Internet Publishing with Microsoft Word 7.0; With CDROM With CDROM
Connie Dunn / Paperback / Published 1998
Our Price: \$29.95 (Not Yet Published)
Read more about this title...
Internet Quick Reference Guide
Glenn Davis / Paperback / Published 1997
Our Price: \$9.60 - You Save: \$2.40 (20%) (Back Ordered)
Read more about this title...

(c) Amazon

Figure 10: Excerpts from three Web sources used in our experiments.

We collected sample pages from our three sources and applied the top-down and bottom-up extraction strategies to them. A single example object was provided for each source, as illustrated in Figure 11.

Using this single example object for each source, we have applied our strategy to extract object from the sample pages we collected. The results are illustrated in Table 1. The sample pages from CDNow contain 219 retrievable objects which were all recognized and extracted by both strategies. The reason is that these objects present a simple inherent structure which is preserved in all pages. The sample pages from Travelocity contain 162 retrievable objects. While the bottom-up strategy is able to recognize and extract all of them, the top-down strategy fails to recognize 21% of them (i.e., 129 or 79% of the objects are retrieved). The reason is that although the implicit object structure occurs in all pages, for some of them one of the attributes is not always present. The sample pages from Amazon contain 89 recognizable objects. While the bottom-up strategy extracts 95% of them, the top-down strategy presents a poor retrieval performance and is able to recognize only 28% of the objects. The main reason is that the objects might now have different attributes which are missing. While the performance of the top-down strategy can be improved by increasing the number of example patterns, there is not much motivation to do so because the bottom-up strategy is already so superior.

Consider again the sample pages from Amazon and assume that the (implicit) objects on those pages are sorted

(a) CDNow

(b) Travelocity

(c) Amazon

Figure 11: Example Objects as specified through the DEByE GUI.

Source	Total	TD	BU
CDNow	219	219 (100%)	219 (100%)
Travelocity	162	129 (79%)	162 (100%)
Amazon	89	25 (28%)	85 (95%)

Table 1: Number of objects retrieved by our top-down (TD) and bottom-up (BU) strategies for three Web sources.

according to their order of appearance. Consider this sorting as a ranking of the objects from these Amazon pages. Given this unusual ranking of the objects, we can plot curves of precision and recall [8] (in 11-standard recall levels) for the results of our bottom-up and top-down strategies. The curves are plotted as follows. Let N be the total number of objects. We traverse the ranked objects starting with the first object and moving towards the last one. At the position n of the ranking, we have traversed n objects. For these n objects, we count the number ℓ of objects that have been properly recognized by each extraction strategy. The precision P indicates which percentage of the objects traversed has been recognized. The recall R indicates which percentage of all objects has been recognized. Thus, the precision P and the recall R are given by: $P = \frac{\ell}{n}$ and $R = \frac{\ell}{N}$. Precision and recall figures for the pages from Amazon are illustrated in Figure 12.

We first notice that the retrieval performance of the top-down strategy deteriorates as it proceeds. This effect indicates that the top-down strategy fails to match objects early on and never recovers. The bottom-up strategy, on the other hand, is able to maintain high precision for levels of recall up to 80%. Then, its precision suddenly drops

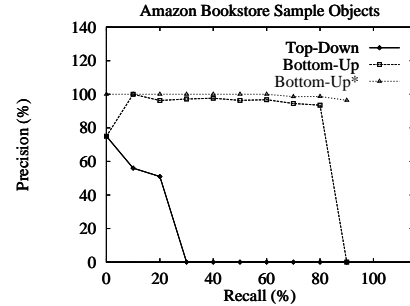


Figure 12: Precision and recall curves relative to sample pages from Amazon.

to zero. This indicates that the objects in the final sample pages have a distinct structure. To deal with this problem, we built a single additional example object (derived from one of the final sample pages). We then rerun our bottom-up strategy using now two example objects (the one that we had originally and the new one just built). The curve labeled Bottom-Up* illustrates the results which indicate a very nice improvement. In fact, the levels of precision are now very close to 100% for the various recall levels.

For the second part of our experiments, we use Web pages from DB&LP ACM TODS [3]. The example provided is shown in Figure 13. The objects in these pages have a two-

Figure 13: Example Object as specified through the DEByE GUI for a sample DB&LP TODS page.

level hierarchical structure as indicated in Figures 1 and 2. To demonstrate the deficiencies of the top-down strategy in dealing with missing pieces of information in multi-level hierarchies, we deleted some component objects (particularly, Date and Pages) from the objects in our set of example pages. Following, we applied the top-down and bottom-up strategies on the modified pages. The results are summarized in Table 2.

Object	Total	TD	BU
1st. Level			
Volume	20	15 (75%)	20 (100%)
Number	20	15 (75%)	20 (100%)
Date	15	15 (100%)	15 (100%)
Edition	20	15 (75%)	20 (100%)
2nd. Level			
Title	76	45 (60.5%)	76 (100%)
Author	188	110 (58.5%)	187 (99.5%)
Page	66	45 (68.2%)	66 (100%)
Article	76	45 (60.5%)	76 (100%)

Table 2: Number of objects retrieved by our top-down (TD) and bottom-up (BU) strategies for the DB&LP Web source.

We observe that, in the first level, there are 20 complex

objects and that 5 of them do not include a Date component. Because of that, these 5 objects are not retrieved by the top-down strategy. Additionally, the top-down strategy also fails to recognize second level components (generated by a decomposition operation) even after properly recognizing the first level components. The bottom-up strategy, on its turn, is able to compose complex objects, even when some of their component objects are missing in the Web source. As a result, it presents a very nice extraction capability.

7 Conclusions

We have studied the problem of extracting semi-structured data from Web pages. Our approach is innovative because it is based solely on a couple of examples provided by the user. Further, it is highly adaptive and deals nicely with objects missing attributes or presenting the attributes out of order.

We showed that example-based data extraction leads to two basic types of strategies: a top-down strategy and a bottom-up strategy. We presented both strategies and discussed the trade offs between them. Through experimentation, we demonstrated that the bottom-up example-based strategy is very effective and retrieves almost all objects present in sample pages obtained from distinct sources. We also notice that our approach is not tied to any specific formatting system (e.g. HTML, XML, LaTeX, etc.). Instead, it takes advantage of any markups surrounding the data of interest.

References

- [1] Amazon.com Web Site. <http://www.amazon.com>.
- [2] CDnow Web Site. <http://www.cdnw.com>.
- [3] DB&LP Site. <http://www.informatik.uni-trier.de/~ley/db/>.
- [4] Travelocity Site. <http://www.travelocity.com/>.
- [5] ADELBERG, B. NoDoSE - A Tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (Seattle, Washington, 1998), pp. 283–294.
- [6] ASHISH, N., AND KNOBLOCK, C. Wrapper Generation for Semi-structured Internet Sources. *ACM SIGMOD Record* 26, 4 (1997), 8–15.
- [7] ATZENI, P., AND MECCA, G. Cut & Paste. In *Proceedings of the ACM Symposium on Principles of Database Systems* (Tucson, Arizona, 1997), pp. 144–153.
- [8] BAEZA-YATES, R., AND RIBEIRO-NETO, B. *Modern Information Retrieval*. Addison-Wesley, Harlow, England, 1999.
- [9] BUNEMAN, P. Semistructured Data. In *Proceedings of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems* (Tucson, Arizona, 1997), pp. 117–121.
- [10] CALLAN, J. P. Passage-Level Evidence in Document Retrieval. In *Proceedings of the ACM SIGIR Conference on Information Retrieval* (Dublin, Ireland, 1994), pp. 302–309.
- [11] CHAWATHE, S., GARCIA-MOLINA, H., HAMMER, J., IRELAND, K., PAPAKONSTANTINOY, Y., ULLMAN, J., AND WIDOM, J. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proceedings of IPSJ Conference* (Tokyo, Japan, 1994), pp. 7–18.
- [12] COWIE, J., AND LEHNERT, W. Information Extraction. *Communications of the ACM* 39, 1 (1996), 80–91.
- [13] EMBLEY, D. W., CAMPBELL, D. M., JIANG, Y., NG, Y., SMITH, R. D., LIDDLE, S. W., AND QUASS, D. W. A Conceptual-Modeling Approach to Extracting Data from the Web. In *Conceptual Modeling - ER'98* (Berlin, 1998), T. W. Li, S. Ram, and M. Lee, Eds., Springer Verlag, pp. 78–91.
- [14] EMBLEY, D. W., CAMPBELL, D. M., LIDDLE, S. W., AND SMITH, R. D. Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents. In *Proceedings of the International Conference on Information and Knowledge Management* (Bethesda, Maryland, 1998), pp. 52–59.
- [15] GRUSER, J., RASCHID, L., VIDAL, M. E., AND BRIGHT, L. Wrapper Generation for Web Accessible Data Sources. In *Proceedings of the Third International Conference on Cooperative Information Systems* (New York City, New York, 1998), pp. 14–23.
- [16] HAMMER, J., GARCIA-MOLINA, H., NESTOROV, S., YERNENI, R., BREUNIG, M., AND VASSALOS, V. Template-Based Wrappers in the TSIMMIS Experience. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (Tucson, Arizona, 1997), pp. 532–535.
- [17] KASZKIEL, M., AND ZOBEL, J. Passage Retrieval Revisited. In *Proceedings of the ACM SIGIR Conference on Information Retrieval* (Philadelphia, USA, 1997), pp. 178–185.
- [18] PYREDDY, P., AND CROFT, W. B. TINTIN: A System for Retrieval in Text Tables. In *Proceedings of the Second ACM International Conference on Digital Libraries* (1997), pp. 193–200.
- [19] RIBEIRO-NETO, B., LAENDER, A. H. F., AND DA SILVA, A. S. Top-down Extraction of Semi-Structured Data. In *Proceedings of the Sixth String Processing and Information Retrieval Symposium* (Cancun, Mexico, 1999), pp. 176–183.
- [20] SILVA, E. S. Extraction of Semi-Structured Data Based on Examples. Master's thesis, Department of Computer Science, Federal University of Minas Gerais, 1999. In portuguese.
- [21] SODERLAND, S. Learning to Extract Text-based Information from the World Wide Web. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining - KDD-97* (Newport Beach, California, 1997), pp. 251–254.
- [22] ZLOOF, M. M. Query-by-Example: A Data Base Language. *IBM Systems Journal* 16, 4 (1977), 324–343.