

# Automatic Repairing of Web Wrappers

Boris Chidlovskii

Xerox Research Centre Europe, Grenoble Laboratory, France  
6, Chemin de Maupertuis, F-38240 Meylan

chidlovskii@xrce.xerox.com

## ABSTRACT

We study the problem of automatic repairing of wrappers for Web information providers. Majority of Web wrappers use “hooks” or “landmarks” to find and extract relevant information from Web pages and such wrappers often become inoperable when the page structure is changed. The solution we propose in this paper extends conventional forward wrappers with alternative classifiers built using content features of extracted information and wrappers processing pages backward. We report some preliminary results of the information extraction recovery and wrapper repairing for a set of real Web provider changes.

## 1. INTRODUCTION

World Wide Web represents a tremendously rich source of information and integrating Web data into user applications has become a common practice. Applications use special software components called *wrappers* that recognize and extract information from Web pages, both static ones and those generated dynamically by CGI or Java scripts upon user requests.

Two main problems relevant to wrapping Web data are the *wrapper generation* and *wrapper maintenance*. Over last years, a considerable progress has been achieved in the reliable wrapper generation and powerful methods have been proposed for the fast induction of wrappers from labeled samples [3, 7, 10]. Instead, little work has been done on the wrapper maintenance when providers change mark-up and structure of pages.

Wrappers are combinations of skipping, parsing and extraction rules; these rules tell an HTML parser how to extract and label PCDATA strings in a page, and it is natural that most rules are tightly linked to the mark-up and structure of provider pages. None of existing wrapper induction methods assumes that changes can happen to the pages; as consequence, wrappers become brittle when, for some reasons, the page mark-up or structure is changed. In such a situation,

wrapper often fails to find out specific “hooks” or “landmarks” in a page, becomes inoperable and can not complete the information extraction. It is often easier to re-learn a wrapper again than to repair a broken one. However, the re-learning requires the user intervention that is not always possible.

The wrapper maintenance is challenging in the case when pages in question undergo massive and sweeping modifications. Fortunately, a much more frequent case is that of *concept shift*, when pages of a given provider undergo some *local and small changes*. It is therefore important to develop methods to generate wrappers with integrated recovery and repairing components capable to recover, automatically when possible, from small changes.

The problem of wrapper maintenance includes several levels of difficulty. The simplest level is the detection of page changes. Kushmerick in [8] proposed a solution that analyzes the page and extracted information and detects the page change with a given accuracy. When the change is detected, the designer is notified; then she re-learns the wrapper from samples of the new format. Knoblock et al. [6] developed a method for wrapper repairing in the case of small mark-up change; it detects the most frequent patterns (like starting or ending words) in labeled strings; these patterns are searched in a page when the wrapper is broken.

On-line learning in domains where target concepts depend on some hidden context and may change over time (financial prediction, medical diagnosis, network performance, etc.) has been studied in [4, 11]. The general approach is such a drifting environment consists of keeping only current trusted examples and hypotheses, storing concept descriptions and reusing them if a previous context reappears. Unfortunately, this approach can not be used in the case of broken wrappers, where no trusted example and hypothesis remain available after the change.

The problem of wrapping HTML pages will be vanishing with a wider employment of XML as lingua franca for data exchange and integration. However, the current expansion of XML remains essentially limited to business-to-business solutions, while the HTML exchange remains and will remain at least for some time the principal way of data exchange with thousands of Web information providers.

In this paper, we address the problem of automatic wrap-

per maintenance, under the assumption of concept shift or small changes. In addition to conventional wrappers, which are essentially grammatical views of pages in questions, we build alternative (and redundant) views, using content features of extracted information. Then we combine the two alternative views, the goal of such a combination is two-fold. First, the alternative feature-based view helps to validate information extracted by the grammatical view (wrapper). Second, when the wrapper fails to extract information, the alternative view is used to resume the reliable information extraction. With a given accuracy, the wrapper detects if it can repair the wrapper itself or it should address to a designer for the manual repairing.

The complexity of information extraction conducted by a wrapper can vary from a simple one-slot extraction to a complex list-like and nested one. The former is typical in the case of extracting book prices or stock values. The later is typical for extracting answers from search engines or price comparison sites. In the following sections, we test the proposed automatic repairing method on wrappers of different types.

## 2. LANDMARK AND GRAMMAR WRAPPERS

The majority of Web wrappers has a special, so called landmark-based structure. A landmark wrapper is a combination of *skipTo*-rules with extraction rules which guide the information extraction. Extraction rules are based on *landmarks* which are groups of consecutive tokens that enable a wrapper to locate the start and end of an item within a page [7, 10]. For example, rule *skipTo*(**<b>**) skips every token until the parser finds landmark **<b>**.

Landmark wrappers work well in the case of one-slot extraction and extraction from pages with rather simple structure like tables. For wrapping pages with complex structure and multiple ambiguous choices, more powerful methods of grammatical [2] or transducer induction [3, 5] can be invoked. Both landmark and grammar wrappers fit the general grammatical approach and can be eventually presented as finite-state automata or transducers. For landmark wrappers, transitions in such automata should allow negated labels for *SkipTo*-rules, like ‘not **<b>**’. On the other hand, any grammar wrapper can be automatically converted into a landmark one [1].

### 2.1 Wrapper example

As an example, we consider a wrapper for the Database and Logic Programming site<sup>1</sup> (DBLP) that provides bibliographic information on computer science publications. In February 2001, the format for answers to title-relevant queries underwent some changes; an answer sample *before the change* is shown in Figure 1 (and corresponding HTML source in Figure 2). Each answer item on the original page contains a number, title, one or more authors, conference, pages and possibly a hyper-link to the electronic edition. A wrapper for this initial format has been generated by using the Iwrap toolkit [1] developed at Xerox Research Centre Europe. The information extraction conducted by a wrapper consists of labeling all PCDATA strings in a page either

<sup>1</sup><http://www.informatik.uni-trier.de/~ley/db/index.html>.

with one of above labels or with label none, when the string is not extracted. The schema of the information extracted by the wrapper can be expressed as a regular expression, `DBLPSchema:=(number, ee?, author+, title, conference, pages)*`.

## Search Result

Query: title = "collaborative filtering"

1	EE	Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, John Riedl: An Algorithmic Framework for Performing Collaborative Filtering. SIGIR 1999: 230-237
2	EE	Dhruv Gupta, Mark Digiovanni, Hiro Narita, Ken Goldberg: Jester 2.0: Collaborative Filtering to Retrieve Jokes (demonstration abstract). SIGIR 1999: 333

Figure 1: DBLP answer example before change.

```
<table border=1><tr>
<td align="right" bgcolor="#CCCCFF">1</td>
<td bgcolor="CCFFCC">
<a href="http://p230-herlocker/">EE</a></td><td>
<a href="http://Herlocker:Jonathan_L.html">
Jonathan L. Herlocker</A>,
<a href="http://Konstan:Joseph_A.html">
Joseph A. Konstan</A>,
<a href="http://Borchers:Al.html">Al Borchers</A>,
<a href="http://Riedl:John.html">John Riedl</a>
: An Algorithmic Framework for Performing
Collaborative Filtering.
<a href="http://sigir99.html#HerlockerKBR99">
SIGIR 1999</a>
: 230-237
</td></tr>
```

Figure 2: HTML source fragment of the DBLP answer example.

The DBLP wrapper induced from labeled examples can be represented as a set of extraction rules (see Table 1) where each rule is a pair (*prefix, label*) and prefixes play a role similar to landmarks.<sup>2</sup> In the general case, wrapper *W* scans a file from the beginning to the end and applies the extraction rules as follows. For a current PCDATA string *t*, prefix *S* of *t* contains all tokens from the beginning till *t*, with all previous PCDATA strings replaced with their labels. Wrapper compares then prefix *S* to prefixes in extraction rules. Prefix *S* matches a rule prefix *s*, if *s* is a suffix of *S*, *S* = *uv*, for some string *u*. In the match is found, string *t* is extracted with the corresponding label. If no exact rule is found, the wrapper runs in an *error*.

<sup>2</sup>In Iwrap, extraction rules may actually contain, in addition to prefixes, conditions on PCDATA strings or their fragments.

Before change		After change
Prefix	Label	
<TITLE>	none	
<h1>	none	
<hr>	none	
</A>	none	
<TD>	number	disappears
<TD>-<a>	ee	disappears
<td>	ee	disappears
<td>-<a>	author	<li>-<a>
<td>-<A>	author	<li>-<A>
none-<a>	author	
author-</a>	title	
title-<a>	conference	
conference-</a>	pages	

Table 1: Extraction rules in DBLP wrapper: before and after the change.

Labeling a PCDATA string may depend on labels of previous strings. Prefix <td> for label ee in the DBLP wrapper is label-independent; any PCDATA string preceded with tag <td> will be labeled as ee. Instead, two prefixes ‘‘author-</a>’’ for title and ‘‘conference-</a>’’ for pages are label-dependent. If tag </a> precedes a current string, then it will be labeled as title if the previous string is author and as pages if the previous string is conference.

All rules in Table 1 are deterministic and no two rules may have the same prefix (prefixes are case-sensitive). All prefixes are minimal and optimal, shortening any prefix would make the rule set nondeterministic. The longest prefix for DBLP is 2; in more complex cases, prefixes can be much longer. A method for detecting optimal and minimal prefixes for extraction rules is described in [2] and an efficient implementation is presented in [1].

An important advantage of a grammar wrapper over a landmark grammar is that extraction rules are independent: if any of extraction rules fails, it does not necessarily result in failing any other rules, which can be a case in landmark grammars where the applicability of certain rules depend on the success of previous ones.

The same DBLP sample page *after the change* is given in Figure 3. The page change concerns both mark-up and structure. The mark-up change is in replacing the <table> element containing <td> sub-elements with a <ul>-list where items are separated by <li> tags. The structure change is in disappearing all strings labeled number and ee. As for the extraction rules (see Table 1), not only rules for number and ee disappear, but 2 of 3 rules for author change, too. The only rule for author that does not change (with prefix ‘‘none-<a>’’) refers to the extraction of second, third, etc. author of a given paper.<sup>3</sup>

### 3. WRAPPER MAINTENANCE

<sup>3</sup>PCDATA strings labeled as none and preceding all authors but the first one, are *comma’s* separating authors; see Figure 2. The rule for these none strings does not change either.

## Search Results for ‘collaborative filtering’

- Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, John Riedl: An Algorithmic Framework for Performing Collaborative Filtering. SIGIR 1999: 230-237
- Dhruv Gupta, Mark Digiovanni, Hiro Narita, Ken Goldberg: Jester 2.0: Collaborative Filtering to Retrieve Jokes (demonstration abstract). SIGIR 1999: 333

Figure 3: DBLP answer example after change.

When the prefix for a current string does not match any extraction rule, the wrapper runs in an error; this triggers the recovery and maintenance routine. In the following, we will distinguish between two sequential maintenance steps, *information extraction (IE) recovery* and *wrapper repairing*. The IE recovery addresses the resume of information extraction from the page; it is aimed at extracting as much relevant data as possible; yet it does not lead necessary to the wrapper repairing. This last is triggered only if the recovery went successfully and the extraction rules can be consistently updated to match the new page format. If IE recovery is not successful or incomplete, the wrapper can not be repaired automatically and should ask for the human help.

Under the assumption of *small changes* (we revise the assumption later in Section 5), we propose a solution to the problem of IE recovery and possible wrapper repairing, based on generating alternative views of pages in questions, in addition to conventional wrappers. These additional views include (1) label classifiers based on syntactic features of extracted information and (2) backward wrappers with extraction rules for the case when files are scanned from the end to the beginning.

When considering changes in pages, three independent cases should be recognized, namely shifts in context, content and structure. *Context shift* is a change in the page mark-up; putting price values in **boldface** or adding home page links to authors’ names are typical examples. Clearly the context shift does not change the extracted information. *Content shift* refers to changes in content of information extracted by the wrapper. Examples here are replacing abbreviations used for conference (“SIGIR”) with their full title (“ACM Conference on Research and Development in Information Retrieval”) or adding prefix “pp.” in page strings. Finally, *structural shift* is the change in the structure of extracted information. Typical examples are addition of new labels, removal of old ones, order permutations, etc. In the case of DBLP case, the context and structural shifts take place. Context shift is in replacing <table> element with <ul>-list and the structural shift is the removal of two labels, number and ee.

Content shift does not hit the wrapper extraction rules. Instead, two other ones, context and structural shifts, may

make landscape/grammar wrappers inoperable. Using automata terminology, the context shift makes inconsistent transition between states, while structural shift makes inconsistent automaton states themselves.

### 3.1 Information extraction recovery

In the following, we consider the wrapping of HTML pages as a special case of *classification problem*, where each PC-DATA string in an input file should be classified into exactly one class/label of  $L$ . For such a classification framework, we develop alternative and possibly redundant views of pages; these redundant views can be useful for IE recovery in the case of concept shift.

First, we consider a conventional landmark/grammar wrapper as a *partial classifier*, where each label, including none, is characterized by a set of associated extraction rules (see Table 1). When processing a page, the wrapper analyzes a current string with its prefix to correctly classify the string. The grammar classifier is partial, so it runs in an error when no exact rule is found.

When a wrapper can not find a rule for a string, the *basic recovery strategy* would be skipping one or more strings in the file till the first string that does match a rule. We note that skipping strings is different from labeling them with none. So if a string  $t$  is preceded with one or more skipped strings, then the prefix of  $t$  can not match any of label-dependent rules. So, the recovery will skip strings until a label-independent rule is matched.

In the DBLP case (see Figure 3 for the sample page after the change), the wrapper runs in error at the first author of the first item, which prefix ‘`...<li><a>`’ does not match any prefix rule. Using the basic recovery routine, the wrapper will skip the first author, then it will analyze and label the following ‘,’ (comma) string as none, because it fits the label-independent prefix `</A>`. This will resume the extraction and all following authors, title, conference and pages will be extracted in a regular way. A new error will occur again at the beginning of the next answer item, and so on.

In the general case, the majority of rules in a wrapper may have label-dependent prefixes and the recovery by skipping strings till one that matches a label-independent rule may be too generous. To solve the problem, we propose to extend the unique so far grammar classifier with alternative views which can be used to help during the IE recovery.

In the following section, we study an alternative classifier of PC-DATA strings based on their *content features*. Content features reflect the content of extracted and labeled information; these features are both primitive (syntactic) ones like the length, the number of separators; and more advanced (semantic) ones, likely number of nouns or date strings.

## 4. CONTENT FEATURES

In this section, we consider the string classification by their content only. We select a set  $F_C$  of 45 content features for the alternative classifier, these features consist of syntactic and semantic ones. *Syntactic* features are similar to those used in [8]; they are the string length, word counts, density of digits, upper-case and lower-case characters and standard

Site	$ L $	$ DT $	$ F $	Err(%)	$ L_c $
Altavista	6	32	13	17.6	3
Google	6	53	10	24.7	2
Excite	6	17	7	9.6	3
Yahoo	5	40	10	16.7	1
Metacrawler	6	27	<b>17</b>	<b>26.7</b>	1
Go	5	25	8	16.9	2
Deja	5	33	10	14.0	2
CNN	6	12	<b>5</b>	16.1	2
Lycos	7	36	12	22.7	3
Ebay	5	23	8	6.9	3
Average	5.7	29.8	10.0	17.2	2.2
DBLP	7	15	7	8.8	5
ACM Search	7	33	11	3.7	4
IEEE DL	5	27	8	<b>0.0</b>	5
Elsevier	10	39	12	4.2	5
Cora	7	39	10	7.7	3
CSBiblio	9	<b>72</b>	18	7.5	2
Average	7.5	37.5	11.0	5.3	4.0
WallStreet	7	<b>9</b>	<b>5</b>	13.0	4
Amazon	4	25	6	12.1	2
FinTimes	5	17	9	20.9	3
Average	5.3	17.0	6.7	15.3	3.0

Table 2: Classification by content features.

delimiters (comma, semicolon, blank, dot, etc.). *Semantic features* include typed tokens such as proper names, abbreviations, url and time strings and noun phrases.

We have inducted content classifiers for 19 Web information providers; for all of them, at least one concept shift has been detected from June 1999 to June 2001. The providers form three groups. Two first groups are general-purpose and specialized (in computer science) search engines. The first group includes Altavista, Google, Excite, Yahoo, Metasearcher, Go, Deja, CNN, HotBot (Lycos) and Ebay search engines. The second group includes DBLP, ACM, IEEE, Elsevier, Cora and CS-Bibliography search facilities. Wrappers in the two groups extract “multi-slot multi-value” information, that is, the result is a list of items and each item contains a number of (*label, value*) pairs. Instead, the third group contains wrappers performing the “one-item multi-slot” information extraction, such as the stock information from Wall Street and Financial Times cites and book information/prices from Amazon.com. In all experiments reported in this and following sections, we have used decision trees [9] as the underlying learning system. We build decision trees with the help of Borgert’s classification software.<sup>4</sup> For each provider, we have used 10 sample pages to learn a decision tree and 10 test pages to evaluate the classification errors.

Table 2 shows results of classification by content features for all providers. Abbreviations used in the table are the following:  $|L|$  - number of labels extracted by a wrapper, including none;  $|DT|$  - number of nodes in the pruned decision tree  $DT$  generated by Borgert’s package,  $|F|$  - number of features used in  $DT$ ;  $Err$  - classification error of  $DT$ ;  $|L_c|$  - number of labels with all correct rules,  $|L_c| \leq L$ .

<sup>4</sup><http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html>.

When processing pages with the content classifiers, we ignore the HTML tags and use only content features of PC-DATA strings to classify them. As the table shows, content classifiers give up to 26.7% classification error in the worst case (Metacrawler.com). Between three provider groups, the best result is obtained for the second group, where strings extracted with user labels (not none) expose particular syntactic features and/or extracted information represents an important part of the page. Instead, for the first and third groups, extracted information represents a rather tiny part, making it difficult to distinguish between extracted and non-extracted (labeled with none) information, which results in a higher classification error.

On the other hand, although only one of 19 classifiers is perfect, for each wrapper there are some labels having highly accurate rules. Therefore their strings can be correctly identified by observing only their content features. For example, DBLP wrapper has three such labels, namely `number`, `ee` and `pages`, they can be accurately identified by their content (see Figure 1). As an example, the perfect rule for `ee` labels is the following: `Length=2, UpperCase=2, Digits=0`.

## 4.1 Combining grammatical rules and content classifiers

Now, it is natural to extend the basic recovery strategy described in Section 3.1 with content classifiers. First, the content classifier can validate information the wrapper extracts in the regular way. Second, when a wrapper runs into errors on input files, the combined IE recovery will not simply skip the strings with unrecognized prefixes, but will apply the corresponding content classifier in order to label such tokens.

We consider an input HTML page  $P$  as a sequence of PC-DATA strings to be labeled. Wrapper  $W$  contains extraction rules and for a current string  $t$ ,  $W(t)$  returns either a label  $l_w \in L$  if it finds a matching rule or 'error', otherwise. The content classifier  $C$  is generated from the content feature set  $F_C$  of strings in sample pages.  $C(t)$  returns a pair  $(l_c, acc)$  where  $l_c$  is the most probable label for  $t$ ,  $l \in L$  and  $acc$  is the accuracy for  $l_c$ . Similarly,  $C(t, l)$  returns the accuracy of labeling string  $t$  with  $l$ . For perfect rules,  $C$  returns  $acc = 1.0$ .

Algorithm 1 presented below scans page  $P$  from the beginning to the end. First it probes wrapper  $W$  with a current string  $t$ ; if  $W$  finds a matching rule with label  $l_w$ ,  $t$  is labeled with  $l_w$  if  $C$  validates  $l_w$  by observing content features of  $t$ , for some threshold validation value, that is,  $C(t, l_w) \geq thValidate$ . If an error occurs,  $C$  provides the most probable label  $l_c$  for  $t$ . If the accuracy of  $l_c$  is superior to a given threshold value,  $thRecovery$ ,  $t$  is labeled with  $l_c$ , otherwise string  $t$  remains unlabeled. Note that Algorithm 1 scans the file only once.

### Algorithm 1. Information extraction with recovery.

$thRecovery$  := recovery threshold  
 $thValidate$  := validation threshold  
 $P$  := HTML page;  $success$  := true

**for** each string  $t$  in  $P$  **do**

```

 $l_w = W(t)$ 
if  $l_w \in L$  and  $C(t, l_w) \geq thValidate$  then
    label  $t$  with  $l_w$ 
if  $l_w$  is 'error' then
     $l_c, acc = C(t)$ 
    if  $acc \geq thRecovery$  then label  $t$  with  $l_c$ 
    else skip  $t$ ;  $success := false$ 
else skip  $t$ ;  $success := false$ 
return  $success$ 

```

The content classifier  $C$  plays a double role in the extraction and recovery routine. First, it validates labels for strings found by extraction rules. Second,  $C$  provides a candidate label for a string when the wrapper runs in an error. This double role confirms the use of two threshold parameters in Algorithm 1. The validation threshold  $thValidate$  confirms the label choice done by the wrapper, and therefore it is lower than recovery threshold  $thRecovery$  in cases when the wrapper runs in error and labeling decision is made only by the content classifier,  $thValidate < thRecovery$ .

## 4.2 Backward wrappers and multi-scan recovery

The use of content classifiers during the IE recovery helps wrappers to faster resume the information extraction. To further improve the accuracy of IE recovery, we invoke yet another alternative view of pages in questions, namely backward wrappers. In contrast to forward wrappers, backward wrappers scan files from the end to the beginning. We note that backward landmark wrappers already used in [6] for faster learning of wrapper extraction rules.

A backward wrapper has the same structure as the forward one; its extraction rules used optimal and minimal *suffices* to uniquely label PC-DATA strings when a file is scanned backward. Like forward wrappers, a backward wrapper is partial and can run in error when the format changes. However, because of the backward scanning, it would fail at positions different from those where the forward wrapper would fail. Therefore, backward extraction rules can help to complete information extraction in positions where the forward wrapper fails. This leads us to the idea of combining the forward and backward extraction rules during the IE recovery.

The joint use of forward and backward wrappers transforms the recovery procedure from one-pass scan into multi-pass one; moreover during the recovery the direction of the file scan can change one or more times. In the following, direct and backward wrappers are denoted as  $W^{fwd}$  and  $W^{bkd}$ , respectively.  $W^{fwd}(t)$  takes into consideration string prefixes,  $W^{bkd}$  considers string suffixes.

Algorithm 2 below completes the information extraction and recovery performed by Algorithm 1. Algorithm 2 runs when Algorithm 1 returns false and fails to accurately complete the information extraction. Algorithm 2 switches the file scan direction and tries to label not yet labeled strings probing their prefixes and suffices with forward and backward wrappers, respectively. Algorithm stops when none of the strings is labeled during the last scan. The algorithm does not apply content classifier  $C$  for the recovery labeling, but

only for the content verification of newly labeled strings.

**Algorithm 2. IE multi-scan recovery with forward and backward wrappers.**

```

success := false
stillRecovery := true; direction := 'bkwd'
while StillRecovery is true do
  stillRecovery := false
  for each unlabeled string t in P do
     $l_w := W^{direction}(t)$ 
    if  $l_w \in L$  and  $C(t, l_w) \geq th_{Validate}$  then
      label t with  $l_w$ ; stillRecovery := true
    else skip t; success := false
  if stillRecovery is true then change direction
return success

```

**4.3 Wrapper repairing**

The information extraction recovery is triggered by wrapper errors on a changed page; it applies Algorithm 1 and possibly Algorithm 2 to accurately label strings in the page using alternative content classifiers and backward wrappers. In turn, the IE recovery triggers the wrapper repairing if the recovery went well and all strings have been labeled with a given threshold of accuracy. It can then automatically re-label sample pages and use them as input to the automatic re-learning of the grammatical classifier, by using any of existing methods for wrapper induction [3, 7, 10]. If instead the recovery is incomplete and some strings in the page remained unlabeled, no trusted samples can be prepared for automatic re-learning and therefore the wrapper repairing can not be successful.

**5. SOME EXPERIMENTS**

The method of information extraction recovery described in Sections 3.1- 4 has been implemented in the Iwrap project at Xerox Research Centre Europe. We have tested the recovery method for 19 information providers forming three different domains (see Section 4), namely, general-purpose and computer science search engines and stock/price wrappers. For each provider, 10 “before-change” pages have been used for learning extraction rules and content classifiers before the format change and 10 “after-change” pages have been used for testing the recovery routine. Below we report some preliminary results.

First we classify the changes happened to all providers. Among three possible format changes, we are primarily interested in context and structural ones. In the case of content change, the wrapper action is to notify the designer and it does not influence the recovery mechanism. So, we have selected and tested such format changes where context or structural shift took place. For 19 reported format changes, context shifts occurred in all 19 cases, and structural shifts occurred in 11 cases (see Table 3).

When using content classifiers, we applied the perfect classification rules ( $th_{Recovery} = 1.0$ ). For the IE recovery, we have tested three recovery methods, namely, the basic recovery (Section 3.1), the basic recovery with content classifiers (Algorithm 1) and the basic recovery with content classifiers and backward wrappers (Algorithms 1 and 2). For

Site	Cnxt shift	Strct shift	$Err_F$ (%)	$Err_{FC}$ (%)	$Err_{FCB}$ (%)
Altavista	✓	✓	13.5	1.4	1.2
Google	✓	✓	17.5	1.2	0.0
Excite	✓	✓	21.8	0.0	0.0
Yahoo	✓	✓	11.0	0.0	0.0
Metacrawler	✓	✓	18.9	5.2	4.9
Go	✓	✓	7.2	0.0	0.0
Deja	✓	✓	19.8	1.8	0.9
CNN	✓	✓	32.1	3.1	2.1
Lycos	✓	✓	6.1	<b>0.0</b>	0.0
Ebay	✓	✓	20.0	3.1	<b>0.0</b>
Average			16.8	1.6	0.9
DBLP	✓	✓	12.4	0.0	0.0
ACM Search	✓	✓	23.3	0.0	0.0
IEEE DL	✓	✓	11.7	0.0	0.0
Elsevier	✓	✓	<b>5.1</b>	0.0	0.0
Cora	✓	✓	7.7	1.2	0.7
CSBiblio	✓	✓	9.1	<b>0.0</b>	<b>0.0</b>
Average			11.5	0.2	0.1
WallStreet	✓	✓	42.8	14.2	<b>14.2</b>
Amazon	✓	✓	<b>50.0</b>	<b>25.0</b>	0.0
FinTimes	✓	✓	40.0	<b>0.0</b>	<b>0.0</b>
Average			44.3	13.1	4.7

**Table 3: Recovery results for changed information providers.**

each method, we measured the *recovery error* which is the percentage of strings remain unlabeled.

Table 3 summarizes the results of information extraction recovery for all providers and all recovery methods. Abbreviations used in the table are as follows:  $Err_F$  - recovery error for the basic recovery by forward extraction rules;  $Err_{FC}$  - recovery error for the recovery by forward wrappers and context classifier;  $Err_{FCB}$  - recovery error for the recovery by forward and backward wrappers and context classifier.

As the table shows, the basic IE recovery is unsuccessful for all 19 providers, it failed to extract 5.1% to 50% of relevant information. Instead, the extension of basic recovery with content classifiers (Algorithm 1) has been successful for 12 of 19 providers. Moreover, for two more providers, the use of backward wrappers (Algorithm 2) permits to complete the extraction and repair the wrapper.

The most representative are recovery results for the wrappers of the third group. For the WallStreet case, the basic recovery is able to extract 4 elements of 7 and the recovery with the content classifier extracts 2 more elements. In the Amazon case, wrapper extracts 2 elements of 4, and the content classifier and backward wrapper extract one more element each. Finally, for the Financial Times wrapper, the basic recovery finds 3 elements of 5 and the content classifier finds two missing ones.

Finally we note that the recovery routines have been applied to all detected format changes, and this allows us to return to the small change assumption mentioned in Section 3. Actually, the adjective “small” was used mainly for the con-

venience of explanation and not to constrain the proposed recovery routines. The success or failure of the IE recovery is determined by a number of aspects, including the type of changes, their density or sparseness in pages, etc. If all these aspects are aggregated in one notion of “size” of a change, then it appears to be highly correlated to the chance for success: the smaller changes happening to the page, the higher probability of the successful automatic recovery.

## 6. CONCLUSION

We have studied the problem of automatic repairing of Web wrappers under the small change assumption. We proposed a solution based on extending the basic recovery routine provided by conventional forward wrappers, with content-based classifiers and backward wrappers. We have tested the proposed recovery method for change cases in widely used Web search engines and information providers with both simple and complex information extraction. Preliminary recovery results are promising, as the recovery went successfully for 14 cases of 19.

The proposed recovery method copes with all cases of format change, including changes in content, context and structure; this gives an advantage over other methods, for example [6] where context changes has been considered. On the other hand, to verify and validate the recovery method we have tested only those provider changes where context and structural shifts took place. The full and exhaustive testing will include all change cases we have collected since 1999, with any combinations of content, context and structural shifts.

## 7. REFERENCES

- [1] Denis Bredelet and Bruno Roustant. Java IWrap: Wrapper Induction by Grammar Learning. Master’s thesis, ENSIMAG, Grenoble, France, 2000.
- [2] B. Chidlovskii. Wrapper Generation by k-Reversible Grammar Induction. In *Proc. ECAI’00 Workshop on Machine Learning Inform. Extraction*, 2000.
- [3] B. Chidlovskii. Wrapping Web Information Providers by Transducer Induction. Proc. European Conference on Machine Learning,, 2001.
- [4] M. Harries and K. Horn. Learning stable concepts in domains with hidden changes in context. In *Learning in context-sensitive domains Workshop, 13th International Conference on Machine Learning*., 1996.
- [5] C.-N. Hsu and C.-C. Chang. Finite-state transducers for semi-structured text mining. In *Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, 1999.
- [6] Craig A. Knoblock, Kristina Lerman, Steven Minton, and Ion Muslea. Accurately and reliably extracting data from the web: A machine learning approach. *IEEE Data Engineering Bulletin*, 23(4):33–41, 2000.
- [7] N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
- [8] N. Kushmerick. Wrapper Verification. *World Wide Web Journal*, 2000.
- [9] T. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.
- [10] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proc. the Third Intern. Conf. on Autonomous Agents Conference, Seattle, WA*, pages 190–197, 1999.
- [11] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, pages 69–101, 1996.