

Wrapping Web Information Providers by Transducer Induction

Boris Chidlovskii

Xerox Research Centre Europe, Grenoble Laboratory, France
6, Chemin de Maupertuis, F-38240 Meylan
E-mail: childovskii@xrce.xerox.com

Abstract. Modern agent and mediator systems communicate to a multitude of Web information providers to better satisfy user requests. They use wrappers to extract relevant information from HTML responses and to annotate it with user-defined labels. A number of approaches exploit the methods of machine learning to induce instances of certain wrapper classes, by assuming the tabular structure of HTML responses and by observing the regularity of extracted fragments in the HTML structure. In this work, we propose a general approach and consider the information extraction conducted by wrappers as a special form of transduction. We make no assumption about the HTML response structure and profit from the advanced methods of transducer induction, in order to develop two powerful wrapper classes, for samples with and without ambiguous translations. We test the proposed induction methods on a set of general-purpose and bibliographic data providers and report the results of experiments.

1 Introduction

The World Wide Web has become an enormous information resource and a bunch of new applications rely on Web search engines, news, weather or shopping sites, in order to find and deliver information relevant to user needs. The communication with Web information providers is done by retrieving Web pages or sending cgi-requests; in either case, an application faces the problem of understanding a provider response in HTML and extracting and labeling relevant information fragments; special software components dedicated to this task are conventionally called *wrappers* [5, 12].

The manual generation of wrappers is a time-consuming and error-prone task and a number of methods have addressed the automatic wrapper generation [4, 8–11, 13, 14]. These methods invoke the machine learning techniques and define the problem of wrapper generation as induction of instances of a certain wrapper class from labeled samples.

While wrapping one-value or one-slot HTML responses, such as stock values or weather forecast rarely poses serious problems to any learning program, wrapping pages with complex structure often unveils its limitations. As an example, most approaches to wrapping search engines assume the tabular form of answers and provide special treatment for different variations, like missing or multi-valued attributes. However, the

assumption does not hold for search engines with sophisticated res-Ponce structure like Medline or Cora ¹.

The power of a wrapper class is crucial as more powerful classes allows one to successfully wrap more sites. In this work, we propose a general approach which makes no assumption about the response structure and consider the information extraction conducted by wrappers as a special form of transduction. Our approach is inspired by the OSTIA transducer induction method [15] which learns transducers from positive samples and pushes back the output labels to postpone translation conflicts. By analogy, in our methods we learn accurate wrapper extraction rules from positive samples which may have ambiguous labeling.

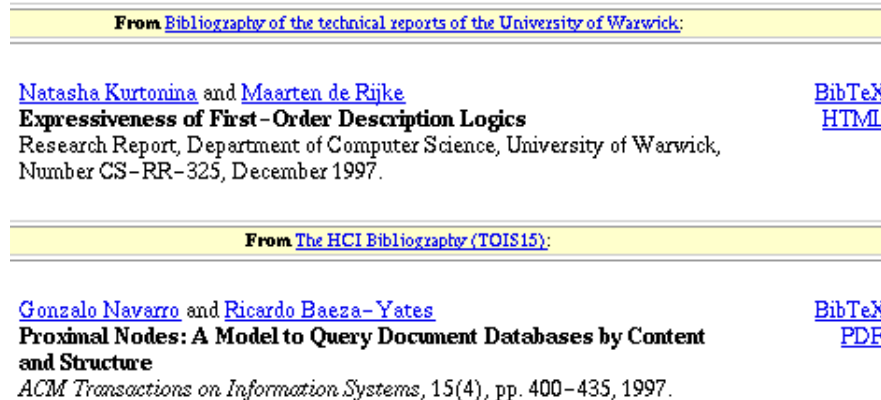


Fig. 1. Sample fragment from CSBiblio site.

In HTML pages with complex structure, extracted information can be complex and thus labels that annotate extracted fragments may have a path-like form. The result of information extraction is *semistructured data*, represented as a tree where each node is labeled and may have a value [5]. The extracted semistructured data can be similar, poor or richer than the structure of original HTML file; its complexity is actually driven by the user's perception and needs. The semistructured data model has a highly expressive power, it permits one to cope with all possible variations in the page structure. Figure 1 shows a sample fragment from the Computer Science Bibliography site.² The extracted data is shown in Figure 2; it has a nested structure where each tuple groups coherent information and tuples are grouped by bibliographic collections.

HTML and XML. The problem of wrapping HTML pages may disappear with the wide use of XML for data exchange. However, the current expansion of XML remains essentially limited to business-to-business solutions, therefore HTML interface will remain, at least for some time in future, the principal way of interaction with thousands of Web information providers.

¹ <http://www.medportal.com>, <http://cora.whizbang.com>.

² <http://liinwww.ira.uka.de/bibliography/index.html>.

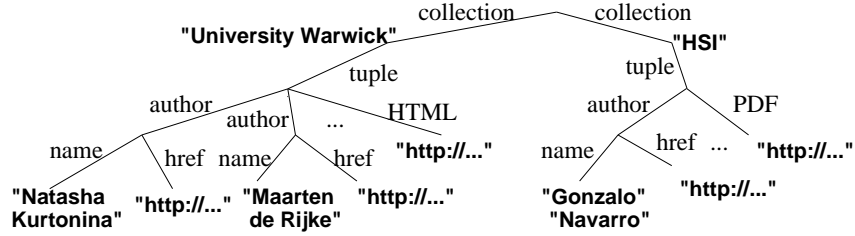


Fig. 2. Extracted semistructured data.

Determinism in wrappers. Applications relying on Web data require from wrappers the delivery of reliable and accurate data. For this reason, the determinism is of the prime interest in the wrapper induction. While the induction methods can be nondeterministic or stochastic, the deterministic information extraction remains a must in the wrapper generation.

2 Transducers

Wrappers parse input HTML strings, extract some tokens and annotate them with user labels. Thus, the information extraction conducted by a wrapper appears similar to transduction of strings of an input alphabet into strings of output alphabet [9]. Below we remind some notions from the formal language theory and discuss the difference between transducers and wrappers.

For any string x over an alphabet Σ , x^r denotes the reverse of x . The concatenation of two strings x and u is denoted xu . The string v is a *prefix* of the string x if and only if there exists a string u such that $vu = x$. A *language* is any subset of Σ^* . $Pr(L)$ denotes all prefixes of L . A *positive sample set* of the language L is a finite set of strings from L .

A *transducer* T is a 6-tuple $T = (\Sigma, H, Q, q_0, F, \delta)$, where Σ and H are input and output alphabets, Q, q_0, F are sets of all, initial and final states, and δ is a map from $Q \times \Sigma$ to $Q \times H$ [7]. Transducer T is deterministic, iff there is at most one rule for any pair (q, a) in δ : if $\delta(q, a) = (q', h')$ and $\delta(q, a) = (q'', h'')$, then $q' = q''$ and $h' = h''$, where $a \in \Sigma, h', h'' \in H, q, q', q'' \in Q$. For an accepted string $x \in \Sigma^*$, transducer T translates x into an output $y = T[x] \in H^*$.

Wrappers and transducers. There are two main difference between wrappers and transducers. The first, obvious one is in the way they produce the output. When a transducer consumes an input token a and executes the transition $\delta(q', a) = (q, h)$, it emits an output symbol h . A wrapper instead outputs the value of token a labeled with h , denoted $a(h)$. In other words, *translation* in transducers corresponds to *labeling* in wrappers. One can easily establish a one-to-one mapping between outputs produced by transducers and wrappers, in the following we will use the notions of translation and labeling interchangeably.

The second, more serious difference between wrappers and transducers is in the way they treat input strings. While transducers cope with accepted input strings and

their translation, the prime interest in wrappers remain the information extraction and correct labeling. As result, we develop a wrapper representation alternative to conventional transducers. Any wrapper is represented as a set of information extraction rules; it neglects the issue of input strings acceptance but addresses the correct and valid information extraction.

HTML tokenization. There are different ways to tokenize HTML files, where elements use opening and closing tags to group element contents and sub-elements. Unfortunately, unlike XML, the HTML standard does not require the proper tag nesting; as consequence, the majority of real-world HTML files are not well-formed. For this reason, we consider a *simple* tokenization of HTML, where alphabet Σ contains all HTML tags (both opening and closing ones) and a special token \mathcal{C} for element contents, $\Sigma = \{\mathcal{C}, \langle \text{html} \rangle, \langle / \text{html} \rangle, \langle \text{title} \rangle, \dots\}$; any HTML file is considered as a string over Σ . For example, an HTML fragment $\langle \text{title} \rangle \text{Home Page} \langle / \text{title} \rangle$ is a sequence of three tokens, $\langle \text{title} \rangle$, \mathcal{C} and $\langle / \text{title} \rangle$, where \mathcal{C} 's value is 'Home Page'.

Elementary information extracted from a tokenized HTML file can be one token or a part of token, like `href` attribute in an `<a>` element or `src` attribute in an `` element. However, in the following, we assume that only element contents (denoted by \mathcal{C}) are extracted from a file, that is, labeled with user-defined labels, like `Author` or `Title`, while all other tokens are labeled with a dummy label $\lambda \in H$.³ This assumption is made only for the sake of simplicity; in the general case, the algorithms can be directly extended to allowing user-defined labels on any tokens from Σ [3].

Example 1. Assume an information provider I generates HTML pages using the following regular structure: $\langle \text{hr} \rangle (\mathcal{C}(\text{Author}) \langle \text{p} \rangle (\mathcal{C}(\text{Title}) \langle \text{hr} \rangle | \langle \text{br} \rangle))^*$. The corresponding transducer T_1 is given by $(\Sigma, H, Q, q_0, F, \delta)$, where $\Sigma = \{\langle \text{hr} \rangle, \langle \text{p} \rangle, \langle \text{br} \rangle, \mathcal{C}\}$, $O = \{\text{Author}, \text{Title}, \lambda\}$, $Q = \{q_0, \dots, q_4\}$, $F = \{q_1\}$ and $\delta = \{\delta(q_0, \langle \text{hr} \rangle) = (q_1, \lambda), \delta(q_1, \mathcal{C}) = (q_2, \text{Author}), \delta(q_2, \langle \text{p} \rangle) = (q_3, \lambda), \delta(q_3, \mathcal{C}) = (q_4, \text{Title}), \delta(q_3, \langle \text{br} \rangle) = (q_4, \lambda), \delta(q_4, \langle \text{hr} \rangle) = (q_1, \lambda)\}$. The transducer is shown in Figure 3; for transparency we omit all occurrences of λ .

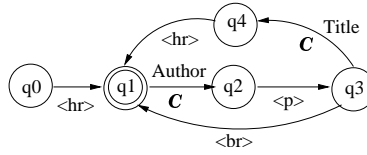


Fig. 3. Transducer T_1 .

³ Those element contents which are not extracted from a file are equally labeled with λ .

4 Deterministic prefix tree transducer

Let $h \in H$ label some occurrences of \mathcal{C} in $s \in S$. We denote as $R(s, h)$ the set of all reverse prefixes for labeled tokens $\mathcal{C}(h)$ in s , $R(s, h) = \{s_1^r | s = s_1 \mathcal{C}(h) s_2\}$. The union of reverse prefixes for h in all $s \in S$ gives the set $R(h) = \{\cup_{s \in S} R(s, h)\}$.

In Example 2, the reverse prefix sets for labels **Title** and **Author** are the following:
 $R(\mathbf{Author}) = \{\langle \text{hr} \rangle, \langle \text{hr} \rangle \mathcal{C}(\mathbf{Title}) \langle \text{b} \rangle \mathcal{C}(\mathbf{Author}) \langle \text{hr} \rangle, \langle \text{br} \rangle \langle \text{p} \rangle \mathcal{C}(\mathbf{Author}) \langle \text{hr} \rangle\}$
and $R(\mathbf{Title}) = \{\langle \text{p} \rangle \mathcal{C}(\mathbf{Author}) \langle \text{hr} \rangle\}$.

The following proposition establishes an important relationship between the determinism of a prefix tree transducer and the emptiness of intersection of reverse prefix sets.

Proposition 1 *For a sample set S , the corresponding prefix tree transducer $PTT(S)$ is deterministic iff for any pair of different labels h and h' in S , $R(h) \cap R(h') = \emptyset$.*

The proposition provides a general mechanism for the deterministic information extraction in wrappers induced from samples. Assume the proposition holds for a given set S , reverse prefix sets are built for all labels in S and we want to extract information from a new sample $x \in \Sigma^*$. Then, for any occurrence of \mathcal{C} in x , it is sufficient to build its reverse prefix, comparing it to all sets $R(h)$ will uniquely identify the label for \mathcal{C} .

Proposition 1 uses the full reverse prefixes for labels in S . However, full reverse prefixes is of little practical use as they can require long learning and huge sample sets. Below we propose *minimal prefix sets* as an alternative to full prefix sets. This alternative representation, on one side, preserves the the relationship established by Proposition 1 and, on other side, is compact and efficient in use.

4.1 Minimal prefix index

For a labeled sample $s = (x, y)$, k leading labeled tokens of s is called *k-prefix* and denoted by $s[k] = (x[k], y[k])$; if $len(s) < k$, $s[k] = s$. Given the prefix set $R(h)$ for label h in S , the set $R_k(h) = \{s[k] | s \in R(h)\}$ is called the *k-prefix set* of h in S , $k = 0, 1, 2, \dots$. For two distinct labels h and h' , the *minimal prefix index* is given by $k(h, h') = \min\{k | R_k(h) \cap R_k(h') = \emptyset\}$.

Example 3. For the sample set S_1 in Example 2, we obtain $k(\mathbf{Author}, \mathbf{Title}) = 1$. Indeed, since $R_1(\mathbf{Author}) = \{\langle \text{hr} \rangle, \langle \text{br} \rangle\}$ and $R_1(\mathbf{Title}) = \{\langle \text{p} \rangle\}$, we obtain that $R_1(\mathbf{Author}) \cap R_1(\mathbf{Title}) = \emptyset$.

For the entire set S , we calculate the *minimal prefix index* as the maximum among $k(h, h')$ for all label pairs in H : $k_S = \max\{k(h, h') | h \neq h'\}$. The value k_S establishes the upper bound on the length of the reverse prefixes to deterministically label any \mathcal{C} occurrence in S .

Minimal prefix set. The way we obtain the minimal prefix index k_S shows how to replace the full prefix sets $R(h)$ for all labels in S with minimal prefix sets, denoted $MR(h)$. The definition for minimal prefix sets is given below; in addition to the feature established in Proposition 1, called separability, such sets should satisfy two additional conditions, namely completeness and minimality.

Definition 1. The minimal (reverse) prefix set MR for label h in a sample set S is such a set of reverse prefixes for which three following features hold:

Separability: $\forall h' \neq h$ in S : $MR(h) \cap R(h') = \emptyset$.

Completeness: for any reverse prefix s in $R(h)$, there exists a prefix $p \in MR(h)$ such that p is a prefix of s .

Minimality: $\forall p \in MR(h)$, replacement of p with any proper prefix of p results in loosing the separability feature.

For any h , $MR(h)$ is obtained from $R(h)$ by replacing any element of $R(h)$, first with its k_S -prefix which satisfies the separability and completeness requirement above. Then, we can try to reduce the prefix further, till the minimality is reached. There exists an incremental procedure which constructs the minimal prefix sets for all labels in a sample set in polynomial time [2]. For the sample set S_1 , we obtain the minimal prefix index k_S equals 1, and $MR(\text{Author}) = \{<hr>,
\}$ and $MR(\text{Title}) = \{<p>\}$.

5 Nondeterministic prefix tree transducer

In the previous section, we have studied the case when the prefix tree transducer built from sample set S is deterministic. However, as we mentioned, $PTT(S)$ can be non-deterministic. In such a case, Proposition 1 does not hold and analysis of token prefixes is insufficient to disambiguate the token labeling. In this section, we extend the prefix-based analysis to suffixes as well and propose a wrapper induction method that copes with non-determinism in sample sets.

Assume that two annotated samples $s^1, s^2 \in S$ start with the prefixes $u\mathcal{C}(h)$ and $u\mathcal{C}(h')$, respectively, where h and h' are different labels of H . Any tree transducer built from set S will have a non-deterministic choice when trying to label the token \mathcal{C} , since samples s^1 and s^2 exemplify the different labeling of \mathcal{C} , with h and h' , respectively.

Though S can be nondeterministic, we still assume that S is *consistent*, that is, there are no two different samples $s^1 = (x^1, y^1)$ and $s^2 = (x^2, y^2)$ in S , such that $x^1 = x^2$ but $y^1 \neq y^2$.

We will say that two distinct occurrences $\mathcal{C}(h)$ and $\mathcal{C}(h')$, $h, h' \in H$ *conflict* in sample set S , if $R(h) \cap R(h') \neq \emptyset$. As an example, consider the transducer T_2 in Figure 5. It has a nondeterministic choice at state q_1 , where token \mathcal{C} can be labeled as either as **Author** or **Title**. Consider an example of annotated sample set corresponding to this transducer.

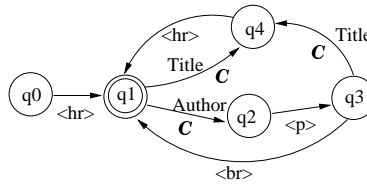


Fig. 5. Nondeterministic transducer T_2 .

Example 4. Let the label set H include labels `Author` and `Title` and the annotated sample set S_2 be $\{ \langle \text{hr} \rangle, \langle \text{hr} \rangle \mathcal{C}(\text{Author}) \langle \text{p} \rangle \mathcal{C}(\text{Title}) \langle \text{hr} \rangle \mathcal{C}(\text{Author}) \langle \text{p} \rangle \langle \text{br} \rangle, \langle \text{hr} \rangle \mathcal{C}(\text{Author}) \langle \text{p} \rangle \langle \text{br} \rangle \mathcal{C}(\text{Author}) \langle \text{p} \rangle \langle \text{br} \rangle, \langle \text{hr} \rangle \mathcal{C}(\text{Title}) \langle \text{hr} \rangle \mathcal{C}(\text{Author}) \langle \text{p} \rangle \mathcal{C}(\text{Title}) \langle \text{br} \rangle \}$. For the sample set S_2 , we obtain the following inverse prefix sets for `Author` and `Title`: $R(\mathcal{C}, \text{Author}) = \{ \langle \text{hr} \rangle, \langle \text{hr} \rangle \mathcal{C}(\text{Title}) \langle \text{p} \rangle \mathcal{C}(\text{Author}) \langle \text{hr} \rangle, \langle \text{br} \rangle \langle \text{p} \rangle \mathcal{C}(\text{Author}) \langle \text{hr} \rangle, \langle \text{hr} \rangle \mathcal{C}(\text{Title}) \langle \text{hr} \rangle \}$, and $R(\mathcal{C}, \text{Title}) = \{ \langle \text{hr} \rangle, \langle \text{p} \rangle \mathcal{C}(\text{Author}) \langle \text{hr} \rangle, \langle \text{p} \rangle \mathcal{C}(\text{Author}) \langle \text{hr} \rangle \mathcal{C}(\text{Title}) \langle \text{hr} \rangle \}$.

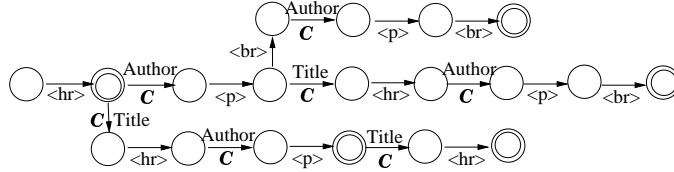


Fig. 6. Prefix tree transducer for S_2 .

These two reverse prefix sets conflict as having the common item `<hr>`. The solution to the prefix conflicts is in extending the analysis, beyond prefixes, to suffixes of input tokens. Indeed, for a labeled token $\mathcal{C}(h)$ in two conflicting samples s^1 and s^2 , the prefix comparison is insufficient, and the conflict resolution might require to lookahead and to compare suffixes of $u\mathcal{C}$ in s^1 and s^2 . Such technique corresponds to postponing the conflict resolution in sub-sequential transducers [15]. If the sample set is consistent, observing both prefixes and suffixes can provide sufficient information to disambiguate labeling in conflict cases. Beyond the case of nondeterminism in sample sets, the analysis of suffixes can be equally helpful when the minimal prefix indexes and sets obtained by the previous method are too large. In such cases, considering *windows* which combine prefixes and suffixes can considerably reduce the number of context tokens needed for the label disambiguation.

Similar to prefix sets, we consider the set of all suffixes for $\mathcal{C}(h)$ in S . Let $\mathcal{C}(h)$ appear in $s \in S$, $s = s_1 \mathcal{C}(h) x'(y')$. The *window* for the given labeled token $\mathcal{C}(h)$ is defined as a pair (s_1^r, x') ; it contains the (labeled) reverse prefix and (not labeled) suffix of $\mathcal{C}(h)$. The set of all windows of $\mathcal{C}(h)$ in s is denoted $W(h, s)$ and the set of all windows of $\mathcal{C}(h)$ in S is denoted $W(h)$: $W(h) = \{\cup_{s \in S} W(h, s)\}$.

The following preposition establishes an important relationship between the consistency of a sample set and label disambiguation.

Proposition 2 *If a sample set S is consistent, then for any pair of labels h and h' in S , we have $W(h) \cap W(h') = \emptyset$.*

Proof. (Sketch) From contradiction, let assume that a given sample set S is consistent but a pair of labels h and h' in S , $W(h) \cap W(h') \neq \emptyset$. Then there exists a window $w = (s, x)$ shared by both sets $W(h)$ and $W(h')$. Therefore S should have two samples s^1 and s^2 such that $s^1 = s \mathcal{C}(h) x(y)$ and $s^2 = s \mathcal{C}(h') x(y')$, and whatever y and y' are, S is not consistent.

5.1 Minimal window index

Proposition 2 gives a general method for the deterministic labeling when sample sets contain ambiguous translations. Similarly to the prefix case, now we introduce minimal window indexes and sets as compact alternative to the entire window sets. We will determine minimal window sets for all labels by observing the k -prefixes and l -suffixes, we can uniquely identify the labeling for any \mathcal{C} occurrence in S .

Given the window set $W(h)$ for $\mathcal{C}(h)$, the set $W_{k,l}(h) = \{(s[k], u[l]) \mid (s, u) \in W(h)\}$ is called the kl -window set of $\mathcal{C}(h)$ in S , $k, l = 0, 1, 2, \dots$. For two distinct labels h and h' , the *minimal window index* is given by $kl(h, h') = \{(k, l) \mid W_{k,l}(h) \cap W_{k,l}(h') = \emptyset\}$, and moreover the sum $k + l$ is minimal among all similar pairs.

For a given sample set S , we calculate the *minimal window index* as the maximal window index $kl_S = (k', l')$ for all pairs of labels in H , where

$$\begin{aligned} k' &= \max\{k \mid (k, l) = kl(h, h'), h \neq h'\}, \\ l' &= \max\{l \mid (k, l) = kl(h, h'), h \neq h'\}. \end{aligned}$$

The value kl_S establishes the upper bound on the lengths of k -prefix and l -suffix that guarantee the uniquely labeling of any occurrence of \mathcal{C} in S .

Minimal prefix set. The minimal window set MW for all h in S are defined in the way similar to the minimal prefix sets. Unfortunately, unlike the minimal prefix sets, there may exist many different minimal window sets for the same set S and finding a globally minimal one is an computationally expensive task. However, there exists a greedy algorithm that build the (locally) minimal window sets for all labels in S .

For the sample set S_2 above, we obtain the (globally) minimal window sets for `Author` and `Title` as follows: $MW(\text{Author}) = \{(\emptyset, \langle \text{hr} \rangle), (\emptyset, \langle \text{br} \rangle)\}$ and $MW(\text{Title}) = \{(\emptyset, \langle \text{p} \rangle)\}$. Finally, $kl(\text{Author}) = kl(\text{Title}) = (0, 1)$.

6 Related work

The wrapper induction problem has been intensively studied over last years [4, 6, 9, 11, 13, 14]. In [11], Kushmerick first identified some simple classes of HTML wrappers which can be efficiently inferred from labeled samples. These classes assume a tabular structure of the response page. The wrapper inference is therefore reduced to the efficient detection of tag sequences preceding each label in such a tabular form. In [13], a wider set of HTML wrappers is considered. Beyond the tabular forms, the method also induces wrapper grammars in cases when some missing labels are they change the appearance order on the response page. Other wrapper classes has been proposed to treat multiple-valued labels, label permutations [9], nested labels in well-formed HTML pages [14] and disjunctions [4, 9, 13]. Some methods [9] are based on transducer-based extractors; however, because the simplifying assumptions about the page structure, none of these methods has however reached the 100% success rate.

7 Experiments and analysis

For experiments, we have selected 16 information providers among about 100 ones generated for Xerox's AskOnce meta-searcher (<http://www.mkms.xerox.com/askonce>). Se-

lected providers cover two large domains, general-purpose search engines (8 sites) and bibliographic data providers (8 sites). They represent a wide range of complexity in response pages, including such different features as varying occurrences and permutations of labels, interleaving response items with advertisements, highlighting query keywords, label nesting, etc.

The complexity of a site is measured by the number of labels, by the size of minimal prefix/window sets (elements of either set are called extraction rules) and by minimal prefix/windows indexes. Complex sites have numerous labels and large MR and MW sets, and consequently, they require more samples for the accurate learning.

The wrappers based on both minimal prefix and minimal window sets have been successfully generated for all sites; in other words, sample sets for all sites are consistent and deterministic. In general, for the selected set, the window-based method outperforms the prefix-based method on 11% in the learning speed, reduces the total number of rules on 6% and number of tokens needed for disambiguation in 1.6 times. Table 1 reports the experiment results for all sites. Abbreviations used in the table are the following:

- H** - total number of labels in H , including λ and **Stop**,
- Sz** - total size of minimal prefix/window sets, $\sum_h |MR(h)|$ or $\sum_h |MW(h)|$,
- RL-m(a), RL-a** - maximal and average size of $MR(h)$ or $MW(h)$,
- k_S, kl_S - minimal prefix/window indexes,
- PL-a, WL-a** - average length of prefixes/windows in $MR(h)$ or $MW(h)$, $PL-a \leq k_S$,
 $WL-a \leq kl_S$;
- L-a** - average number of samples needed to reach 99%-accuracy.

The label set H for all wrappers have been extended with a special label **Stop** which is learned like any other label; it results in halting the information extraction process. Using **Stop** allows wrappers to skip the mute tails of HTML pages and thus to make the minimal prefix/window sets smaller.

For each site, $|S| = 15$ labeled samples have been prepared for the evaluation of wrapper induction process. Accuracy of learning is measured by the percentage of correctly extracted and labeled tokens from a page. For each site, 5 tests have been performed, each test consists in learning the minimal prefix/window sets from $i=1,2,3,\dots$ randomly chosen samples and testing the remaining $(15-i)$ samples against the learned set. The average number (over 5 experiments) of samples needed to learn 99%-accurate prefix/window sets is reported in the table. Here we summarize some important results.

1. The second group of providers appears to be more complex than the first one. *Deja.com* and *Go.com* are the simplest among selected providers, their results have a regular table-like form. The prefix/window sets for both sites contain one rule per each valuable label and the remaining rules, 15 and 16 respectively, cope with label λ used to ignore HTML fragments before and within the answer list.
2. Many sites have a large gap between **RL-m** (referring always to label λ) and **RL-a** values. This gap is particularly important for advertisement-based sites like *Altavista* and *Excite*, while it remains minimal for no-advertisement sites (*Google.com*, *DBLP*, *ACM*⁴ and *Medline*).

⁴ <http://www.informatik.uni-trier.de/~ley/db>, <http://www.acm.org/dl/Search.html>.

3. Cora is an example of search engines that visualize **query keywords** in answers by using a particular font. This splits a unique user label, such as Abstract, in multiple fragments; it results in an important number of rules for Abstract label, in both prefix-based and window-based induction methods.
4. Yahoo.com answers combine category entries and sites. These two sublists have common and different labels. This makes the learning of Yahoo wrapper much longer as compared to any other provider in the first group.
5. Medline differs from other sites by the large number of user labels (21). Although other parameters have values close to average ones, such a large number of labels and their combinations results in the biggest size of the sample set and the longest learning process.
6. The most difficult case takes place with CSBiblio and IEEE ⁵ sites where several labels like references to PDF/PS/HTML copies of a publication often share the same context (see Figure 1); this results in a very large prefix and window indexes (65 and 44 for CSBiblio, 45 and 23 for IEEE). Consequently, large prefixes/windows require a longer learning. Possible solutions for cases like CSBiblio and IEEE are either the normalization of conflicting labels by merging them in one joint label [3] or the extension of purely grammatical inference methods with some non-grammatical components (such as recognizers for explicit strings like 'PDF', 'HTML', etc.) [2, 9, 11].

Site	H	Prefixes						Windows					
		Sz	RL-a	RL-m	PL-a	k_S	L-a	Sz	RL-a	RL-m	WL-a	kl_S	L-a
Altavista.com	7	37	5.3	26	1.5	4	2.1	37	5.3	26	1.5	4	2.1
Google.com	9	29	3.2	11	2.4	4	2.3	27	3.0	11	2.3	3	2.1
Excite.com	7	27	3.9	19	1.4	3	2.0	27	3.9	19	1.3	3	2.0
Yahoo.com	6	30	5.0	14	3.8	20	4.4	29	4.8	14	1.3	4	3.8
Metacrawler.com	9	37	4.1	21	2.2	6	2.6	34	2.8	18	2.0	5	2.2
Go.com	6	21	3.5	16	1.8	8	1.2	19	3.2	14	1.5	4	1.1
Deja.com	6	20	3.3	15	1.6	4	1.1	17	2.8	12	1.4	3	1.1
CNN.com	7	36	5.1	22	1.5	4	2.2	35	5.1	21	1.5	4	2.2
DBLP	8	13	1.6	3	1.8	3	1.6	13	1.6	3	1.8	3	1.6
ResearchIndex	5	28	5.6	17	2.2	7	2.5	24	4.9	15	1.9	4	2.2
CSBiblio	10	32	3.2	17	4.6	65	8.7	32	3.2	17	3.8	44	8.2
ACM Search	9	18	2.0	9	2.1	4	1.6	18	2.0	9	2.1	4	1.6
IEEE DL	7	23	3.3	16	3.8	45	6.5	21	3.0	15	2.7	23	5.0
Elsevier	12	28	2.3	12	2.7	9	1.5	26	2.1	11	2.1	4	1.5
Medline	21	43	2.0	6	2.2	5	9.2	38	1.8	6	2.0	5	8.5
Cora	12	34	2.8	15	2.7	6	3.3	32	2.6	15	2.5	6	3.0
Average	8.8	28.5	3.5	14.9	2.4	12.3	3.3	26.8	3.2	14.1	2.0	7.7	3.0

Table 1. Results of wrapper induction for 16 information providers.

⁵ <http://www.computer.org/search.htm>

8 Conclusion

We have proposed two powerful wrapper classes for mediator and agent Web systems. The wrapper generation based on the transducer induction has been tested with a number of real Web information providers. Experiments have shown a large expressive power of new wrapper classes. Experiments with such sites as IEEE and CSBiblio have been particularly helpful, as they help to establish the limits of prefix-based and window-based methods, and pure grammar-based methods in general. They show that if the perfect information extraction and labeling is required from wrappers, pure grammatical methods should be extended with non-grammatical components.

References

1. Dana Angluin. Inference of Reversible Languages. *Journal of the ACM*, 29(3):741–765, July 1982.
2. Denis Bredelet and Bruno Roustant. Jwrap: Wrapper induction by grammar learning. Master's thesis, ENSIMAG, Grenoble, France, 2000.
3. B. Chidlovskii. Wrapper Generation by k-Reversible Grammar Induction. In *Proc. ECAI'00 Workshop on Machine Learning Inform. Extraction*, 2000.
4. B. Chidlovskii, J. Ragetli, and M. de Rijke. Wrapper Generation via Grammar Induction. In *11th European Conf. Machine Learning, Barcelona, Spain*, 2000.
5. D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3):59–74, 1998.
6. D. Freitag. Information extraction from HTML: Application of a general machine learning approach. In *Proc. AAAI/IAAI*, pages 517–523, 1998.
7. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, N. Reading, MA, 1980.
8. 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.
9. C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semistructured data extraction from the web. *Information Systems*, 23(8), 1998.
10. N. Kushmerick. Wrapper induction; efficiency and expressiveness. In *AAAI'98 Workshop on AI and Information Integration*, 1998.
11. N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
12. N. Kushmerick, D.S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.
13. I. Muslea, S. Minton, and C. Knoblock. Stalker: Learning extraction rules for semistructured, web-based information sources. In *AAAI Workshop on AI and Information Integration*, 1998.
14. 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.
15. J. Oncina, P. Garcia, and E. Vidal. Learning subsequential transducers for pattern recognition interpretation. *IEEE Trans. on Pattern Analysis*, 15:448–458, 1993.