Post-supervised Template Induction for Information Extraction from Lists and Tables in Dynamic Web Sources

Zhongmin Shi
Evangelos Milios
Nur Zincir-Heywood

Technical Report CS-2002-09

November 20, 2002

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

Post-supervised Template Induction for Information Extraction from Lists and Tables in Dynamic Web Sources

Zhongmin Shi, Evangelos Milios, Nur Zincir-Heywood

Faculty of Computer Science,

Dalhousie University,

Halifax, N.S., Canada B3H 1W5

{zincir,eem}@cs.dal.ca

Abstract

Dynamic web sites commonly return information in the form of lists and tables. Although hand crafting an extraction program for a specific template is time-consuming but straightforward, it is desirable to automatically generate template extraction programs from examples of lists and tables in html documents. Supervised approaches have been shown to achieve high accuracy, but they require manual labeling of training examples, which is also time consuming. Fully unsupervised approaches, which extract rows and columns by detecting regularities in the data, cannot provide sufficient accuracy for practical domains. We describe a novel technique, Post-supervised Learning, which exploits unsupervised learning to avoid the need for training examples, while minimally involving the user to achieve high accuracy. We have developed unsupervised algorithms to extract the number of rows and adopted a dynamic programming algorithm for extracting columns. Our method achieves high performance with minimal user input compared to fully supervised techniques.

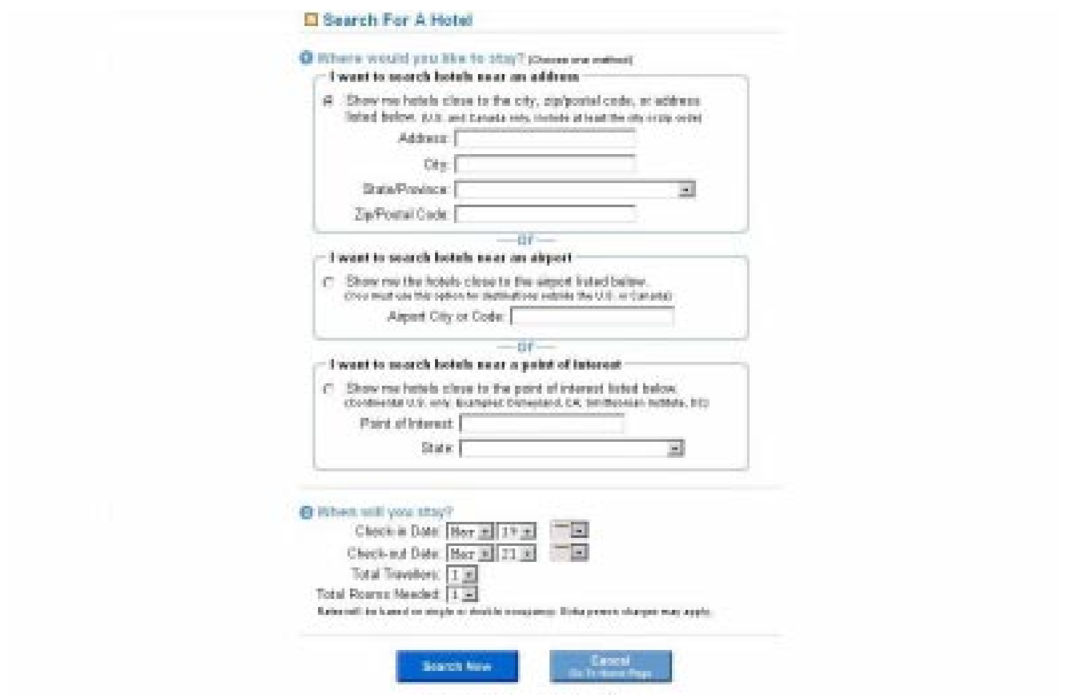
1 Introduction

Dynamically generated web pages composed of a list or table are becoming widely used. The data contained in the list or table is typically extracted from a database on the basis of a user query, as shown in Figure 1. The process is geared towards a human user who employs a web browser to interact with a web site, which is the interface to a database. The process of form filling (to input the query) and viewing the resulting lists or tables is time consuming, and it would be desirable to automate it, especially when this process has to be repeated many times, either to track information over time, or to obtain data for a large number of different queries. If the web site does not change, then it is fairly straightforward to customize an information extraction program to the template of a particular web site. However, in case a large number of different sites needs to be queried or the format of the web page changes, the amount of programming effort required would be prohibitive, and the ability to automatically adapt the template extraction code would be essential.

The World Wide Web has been dominated for a decade by HTML based on a browsing paradigm [6], which is designed for good look-and-feel and easy reading by a human using a Web browser, instead of facilitating the extraction of information by a program. It is therefore difficult to extract information by HTML parsing. Until more structured representations replace, most web clients rely on existing information extraction techniques, typically Web Wrappers [12].

A wrapper is a program that enables a Web source to be queried as if it were a database [9, 8]. Extraction rules used by the wrapper to identify the beginning and end of the data field to be extracted, form an important part of the wrapper. Quick and efficient generation of extraction rules, so called Wrapper Induction, has been an active area of research in recent years [12, 11]. The first wrapper induction system, WIEN [10] is a supervised learning agent, i.e. it requires manually labelled examples with output information, to learn patterns.

A recent wrapper induction algorithm, STALKER, generates high accuracy extraction rules that accept all positive and reject all negative user-labelled training examples [16], and extracts data that complies with these rules with about 80% accuracy on test examples. STALKER uses wildcards and disjunctive rules and therefore has the ability to wrap a larger



Search For A Hotel

Where would you like to stay? (Choose one method)

I want to search hotels near an address

Show me hotels close to the city, zip/postal code, or address listed below. U.S. and Canada only. Include at least the city or zip code

Address:

City:

State/Province:

Zip/Postal Code:

—or—

I want to search hotels near an airport

Show me the hotels close to the airport listed below. (You must use this option for destinations outside the U.S. or Canada)

Airport City or Code:

—or—

I want to search hotels near a point of interest

Show me hotels close to the point of interest listed below. (Continental U.S. only. Examples: Disneyland, CA; Smithsonian Institute, DC)

Point of Interest:

State:

When will you stay?

Check-in Date: Nov 17

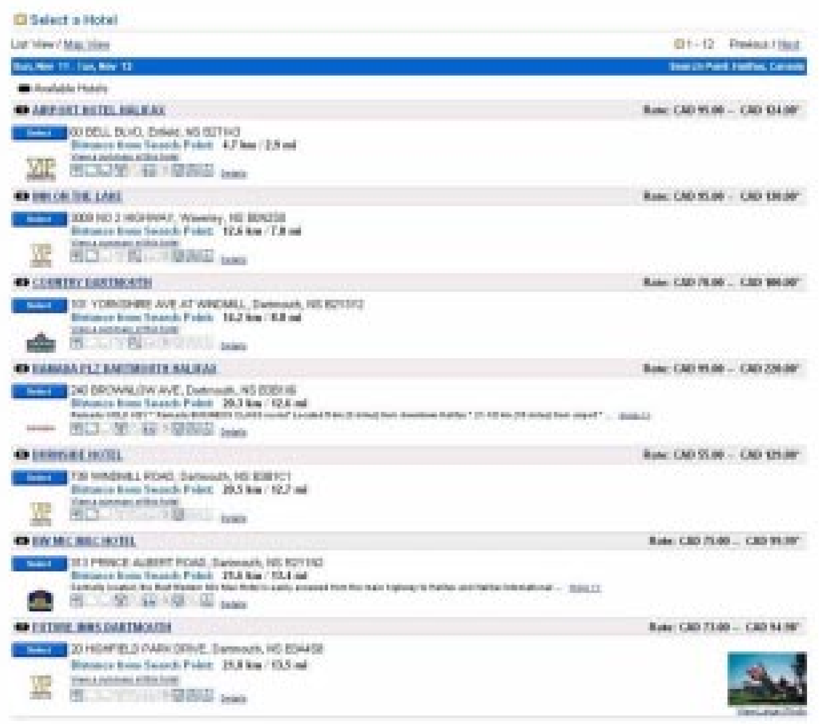
Check-out Date: Nov 21

Total Travelers: 1

Total Rooms Needed: 1

Rates will be based on single or double occupancy. Extra person charges may apply.

a. Inquiry form page



Select a Hotel

List View / Map View — 1 - 12 Previous / Next

Sun, Nov 11 - Tue, Nov 13 — Search Point: Halifax, Canada

Available Hotels

AIRPORT HOTEL HALIFAX — Rate: CAD 95.00 – CAD 124.00*
60 BELL BLVD, Enfield, NS B2T1K3
Distance from Search Point: 4.7 km / 2.9 mi

INN ON THE LAKE — Rate: CAD 95.00 – CAD 130.00*
3009 NO 2 HIGHWAY, Waverley, NS B2N2S8
Distance from Search Point: 12.6 km / 7.8 mi

COUNTRY DARTMOUTH — Rate: CAD 78.00 – CAD 100.00*
101 YORKSHIRE AVE AT WINDMILL, Dartmouth, NS B2Y3Y2
Distance from Search Point: 14.2 km / 8.8 mi

RAMADA PLZ DARTMOUTH HALIFAX — Rate: CAD 95.00 – CAD 220.00*
240 BROWNLOW AVE, Dartmouth, NS B3B1X6
Distance from Search Point: 20.3 km / 12.6 mi

DRIVESIDE HOTEL — Rate: CAD 55.00 – CAD 125.00*
730 WINDMILL ROAD, Dartmouth, NS B3B1C1
Distance from Search Point: 20.5 km / 12.7 mi

BW MIC MAC HOTEL — Rate: CAD 75.00 – CAD 99.00*
313 PRINCE ALBERT ROAD, Dartmouth, NS B2Y1N3
Distance from Search Point: 21.6 km / 13.4 mi

FUTURE INNS DARTMOUTH — Rate: CAD 73.00 – CAD 94.00*
20 HIGHFIELD PARK DRIVE, Dartmouth, NS B3A4S8
Distance from Search Point: 21.8 km / 13.5 mi

b. Inquiry result page

Figure 1: An example of a dynamic web site, www.Travelocity.com[19]. Web clients can submit queries via the web form shown in Figure (a) to the server; figure (b) illustrates the response page including a list of search results.

variety of sources [16] than previous systems, but it still requires manually labelled examples.

To overcome the shortcomings of supervised learning, attention is shifting towards unsupervised learning [12], which needs no manually labelled input. That work proposes a suite of unsupervised learning algorithms, which induce the structure of lists by exploiting the regularities both in the format of the pages and the data contained in them. Some general assumptions are made about the structures of lists and tables with a following four-step approach.

1. Separators are used to partition the web page. The page template is then extracted and the list or table is identified from a set of similar pages.

2. An unsupervised classification algorithm is used to automatically group the web page contents into classes based on separators.

3. Syntactic data patterns, which describe the common start and end of classes [13], are learned by a pattern learning algorithm. It is assumed that data having the same pattern belong to the same column.

4. A grammar induction algorithm is used to build the finite state automaton of the web page. States that represent the same data contents are then merged, and cycles are generated. The longest cycles that correspond to rows are selected [12]

That system was tested on 14 typical examples[12] with about 70% accuracy. It is undoubtedly a significant effort, but the accuracy still needs further improvement. Moreover, it requires several similar web pages to generate the page template in the first step; the general algorithms used, such as DataPro [13] for learning data pattern and AutoClass for unsupervised classification [2, 7], are computationally intensive, a serious problem in Web applications.

Our work aims to improve the performance of information extraction from lists and tables in the web page. Because high-performance unsupervised learning is rather ambitious, the authors have developed a *Post-supervised* learning technique, which employs a suite of unsupervised learning algorithms and minimal user interaction on the results. Our system focuses on:

- achieving approximately 100 percent accuracy
- avoiding user labeled training examples
- minimizing the involvement of the user
- improving list/table identification techniques
- the design for non-programmer users

Figure 2 illustrates the whole system at high level.

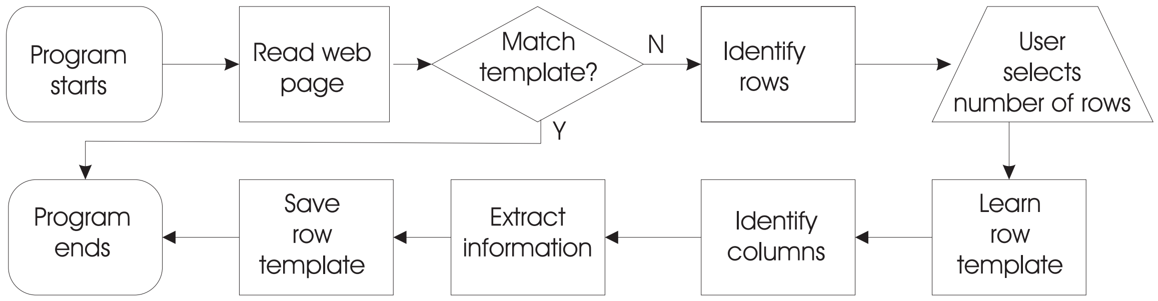


Figure 2: High level flowchart of the system

2 Identifying Rows

The goal of information extraction is to convert displayed information in the dynamic web page back into structured database format. A standard assumption adopted by the authors is that a dynamic web page including lists and tables is generated by a template, which describes the format of each data field and the visual layout of the whole page[12]. The server-side program fills the template with results of a database query submitted by a Web client (browser). Thus, template extraction is a necessary step in this process.

To extract the template, a basic step is identifying common data among a set of dynamic web pages from the same source. Several similar web pages that are usually generated by the same server-side program are required to extract common information among them, in the form of a Page Template [12].

Definition 2.1 (Page template) *A set of strings that the web server uses to automatically generate pages and fill them with the results of database query.*

Since the objective of this work is to identify the main list or table in the web page, data that does not belong to the list or table should be ignored. The focus should be on the template that generates the rows of the list or table, the Row Template. It is assumed that each row in the list or table is generated by a Row Template, which is the main part of the page template.

Definition 2.2 (Row template) *A set of strings that the web server uses to automatically generate rows of lists or tables in the web page.*

Locating the beginning and the end of each row is a necessary step before identifying the row template. The definition of row template suggests that the number of times that some data are repeated in the web page is equal to the number of rows. The row identification procedure is illustrated in Figure 3.

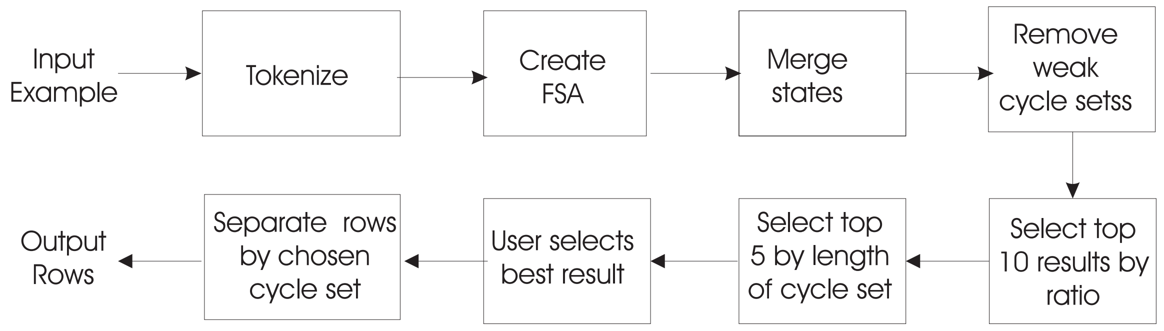


Figure 3: High-level flow chart of the row identification procedure

2.1 Tokenizing

To track repeating data, we split the content of a web page into individual text chunks. This procedure is called tokenizing. The first step in the process is to define special strings that are likely to be found on the boundaries of tokens.

Definition 2.3 (Separator) *Symbols that separate the web page into individual data fields are called separators.*

Since the data fields we are interested in extracting are visually displayed on the web page, it is intuitive that HTML tags should be treated separately from other data, and punctuation characters are often used to separate data fields. Therefore, a separator is defined as one or more consecutive HTML tags, or any punctuation character excluding the set ",.(-)'%" and SPACE, which was selected empirically. Sometimes a dash, "-", is a good separator, but for many frequently encountered data types, such as phone numbers and zip codes, dash is part of data and not a separator [12]. Likewise, a sequence of words that is visually displayed on the web page, e.g., "3009 NO 2 HIGHWAY, Waverley, NS", should not be separated, so SPACE and comma(,) are not regarded as separators.

Definition 2.4 (Token) *A token is a sequence of characters between separators*[1] *in the web page.*

Each token is assigned an index in the web page starting from 0.

Example 1. A sequence of characters:

$$< b > INN\ ON\ THE\ LAKE < /b >< /a >< /td >< td\ align = right > ...$$

is separated into six tokens with indices:

$0 :< b >$

$1 : INN\ ON\ THE\ LAKE$

$2 :< /b >$

$3 :< /a >$

$4 :< /td >$

$5 :< td\ align = right >$

Definition 2.5 (Token Sequence) *A Token Sequence is a sequence of consecutive tokens in the web page. The* length *of a token sequence is the number of tokens it consists of.*

[1]The token includes the separator right after it except for "<", which is included into the token behind it

In example 1, "$< b > INN\ ON\ THE\ LAKE < /b >< /a >< /td >< td\ align = right >$" is called a token sequence.

The assumption, on which the definition of row template is based, can be restated using the above definitions as:

Assumption 1 *All rows contain some common token sequences.*

Example 2. A web page includes two rows like

... $< b > AIRPORT\ HOTEL\ HALIFAX < /b >< /a >< /td >< td\ align = right$ ...

... $< b > INN\ ON\ THE\ LAKE < /b >< /a >< /td >< td\ align = right$ ...

Two common token sequences, $< b >$ and $< /b >< /a >< /td >< td\ align = right$, may be parts of the row template.

2.2 Creating Finite State Automata

.

A grammar induction algorithm [12, 1] is applied to find the repeated data that may correspond to rows. The entire sequence of tokens in the web page is viewed as a string in a language generated by a regular grammar, and the goal is to:

- Construct a Finite State Automaton (FSA) that implements the regular grammar generating the web page.
- Minimize the FSA.
- Learn and use the FSA to recognize the rows.

First a FSA $M = (K, \Sigma, \Delta, s, f)$ of the web page is defined, where:

Σ is an alphabet. Every token in the web page is a symbol of the alphabet.

K is a finite set of states. Each state is between two consecutive tokens.

$s \in K$ is the initial state at the beginning of the web page.

$f \subseteq K$ is the final state at the end of the web page.

Δ, the transition relation, is a subset of $K \times (\Sigma \cup \{e\}) \times K$ [14].

For instance, Figure 4 shows the state diagram of the automaton of Example 2. The minimization procedure consists of state-merging and removal of superfluous transitions.

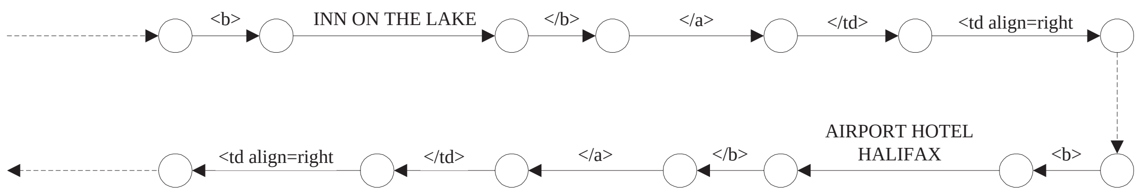


Figure 4: State diagram of automaton of Example 2

Two states, i and j, are merged if their incoming transitions, $\delta_{k,i}(a)$ and $\delta_{l,j}(a)$, correspond to the same symbol a, and at least one of the outgoing transitions, $\delta_{i,m}(b)$ and $\delta_{j,n}(b)$, from each state correspond to the same symbol b. Figure 5 illustrates the automaton of Example 2 after state-merging.

Definition 2.6 *A* ***cycle*** *is a set of consecutive transitions that starts and ends at the same state. A cycle corresponds to a candidate row.*

Definition 2.7 ***Length of cycle*** *is the number of tokens along the cycle.*

Definition 2.8 ***Cycle set*** *is a set of cycles that include at least one common state. It corresponds to the candidate list or table.*

Definition 2.9 ***Overlapping parts*** *are parts of cycles in the cycle set overlapping with each other due to state-merging. They correspond to the common token sequences of candidate rows.*

For a real web page, the automaton is much more complicated than the above example since any repeating tokens will generate cycles, resulting in a large number of cycle sets generated. In order to process all these cycle sets and extract the one that corresponds to the correct rows of the list or table, further processing is required. In the rest of this section, several filters of weak cycle sets are described that greatly enhance the performance of the system. Weak cycle sets are loosely defined as cycle sets that are not very likely to correspond to a row.

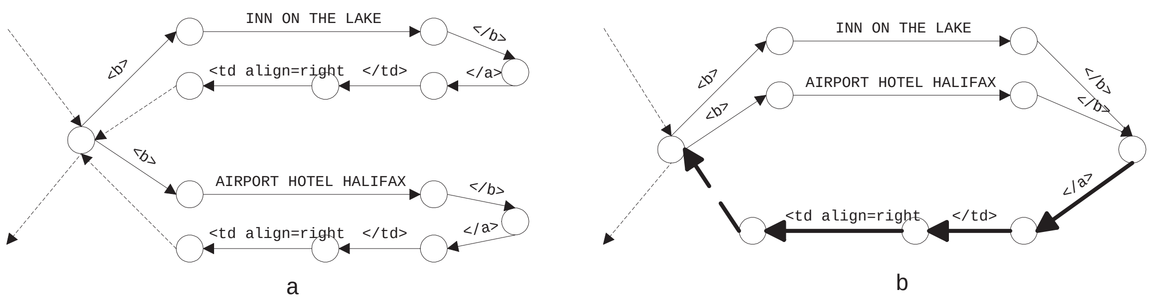


Figure 5: State diagram of state-merged automaton of Example 2. (a) The automaton with only one pair of states merged. (b) The final automaton after full merging. Thicker lines represent overlapping parts of the cycle set.

2.3 Filtering on the basis of variance of cycle lengths in cycle set

The first assumption made is that rows of the same list or table should have similar numbers of tokens. This is a reasonable assumption that applies well to sites generating tables of rows providing the same information about a list of items (for example names, addresses, contact information and pricing of a list of hotels in a city).

Assumption 2 *The number of tokens in a row is close to those of other rows within the same list or table.*

Based on this assumption, a set of cycles should be removed if the lengths of the individual cycles differ greatly from each other. Deviation analysis of a distribution [5] is a general method in statistics to calculate the dispersion degree of a set of data, by normalizing the standard deviation of the data set by its expected value. The coefficient of variation, $Disp(S)$ [17] of a cycle set S is thus defined by the following equation, based on the set L of lengths of cycles in S and assuming that lengths of rows in the list or table are normally distributed.

$$Disp(S) = \sigma(L)/E(L) \tag{2.1}$$

where $\sigma(L)$ represents the standard deviation of L and $E(L)$ means the expectation of L.

In this work, the acceptable dispersion degree is limited to 1, a rather loose limitation. Any cycle set with dispersion degree larger than 1 will be filtered out.

2.4 Filtering based on the length of overlapping cycle parts in cycle sets

Since each cycle in the automaton corresponds to a row, the task of identifying rows requires a procedure of evaluating and distinguishing cycle sets that may correctly represent the table. A possible way is to choose the cycle set with the longest cycles [12]. However, our observations show that the longest cycle criterion is not necessary correct, since some trivial data may generate a long cycle. For instance, some web pages have similar data at the beginning and end, like header and footer. Thus, the longest cycles do not always correspond to rows. Compared to the automaton created by real rows, this kind of weak cycle has the following features:

- Smaller number of cycles in the cycle set, and/or
- Shorter overlapping parts of cycles in the cycle set.

Therefore, to minimize the effect of such cycles, it may be preferable to focus on the number of cycles, and the length of overlapping parts, instead of the total length of cycles in each cycle set. Following this intuition, cycle sets are clustered according to the their number of cycles N into **groups** , and **the sum of lengths $L(N)$ of overlapping cycle parts, i.e. the number of repeated tokens, in each group** is calculated. For example, descriptions of some cycle sets of the example in Figure 1 are shown in Table 1. They are separated into 2 groups corresponding to the value of N, i.e., 12 and 13. Then we calculate $L(N)$ by summing up the length len of cycle sets in each group. Thus, $L(13) = 2 + 5 + 3 + 9 + 4 + 6 + 3 + 3$ and $L(12) = 9 + 8 + 7 + ... + 16 + 15 + 14$. One or more groups are expected to stand out.

An empirical observation related to the number of repeated tokens is that, for a table with n rows, the number of tokens repeated n times is fairly close to the number of tokens repeated $n - 1$ times, but much greater than the number of tokens repeated $n + 1$ times.

To quantify the above observation as a criterion, a group with cycle sets containing N repeated cycles is very likely to correspond to the correct number of rows if the following condition is satisfied.

N	ind	len	N	ind	len
13	690	2	13	880	4
13	866	5	13	992	6
13	868	3	13	995	3
12	999	9	12	1005	3
12	1000	8	12	1018	2
12	1001	7	12	1022	16
12	1002	6	12	1023	15
12	1003	5	12	1024	14

Table 1: Examples of Cycle sets with N equal to 12 and 13 of the example in Figure 1. Each row represents one cycle set. N is the the number of repeated cycles in the cycle set. *ind* is the index of the first token of the cycle set, which indicates the location of the cycle set in the web page. *len* is the length of overlapping parts of the cycle set.

$$L(N-1)/L(N) << L(N)/L(N+1) \tag{2.2}$$

Our system chooses the top 10 groups, ranked by decreasing value of $L(N)/L(N+1)$. This effectively means choosing ten candidate values for the number of rows in the table.

As an example, consider the Travelocity web page [19] with 12 rows shown in Figure 1. Figure 6 shows $L(N)$ for groups of cycle sets with varying N. We observe that the group with smaller number of repeated cycles N usually has larger sum of length of their overlapping parts. However, when looking into the sum of lengths corresponding to the group with 12 repeated cycles, 495, we noticed that it is comparably close to that of the group with 11 repeated cycles, 798, but much larger than 90, the sum of lengths of the group with 13 repeated cycles. Figure 7 shows the **Ratio** $L(N)/L(N+1)$ as a function of N. The group with 12 repeated cycles clearly stands out. Similar patterns are observed in several other web sites. Figure 8 shows the graph corresponding to a web page with 10 rows from `www.whitepapers.com` [20].

2.5 Filtering based on the total number of Tokens in cycle set

The last filtering stage is based on the assumption that, out of the candidate cycle sets, the ones more likely to correspond to the correct number of rows are those that contain the

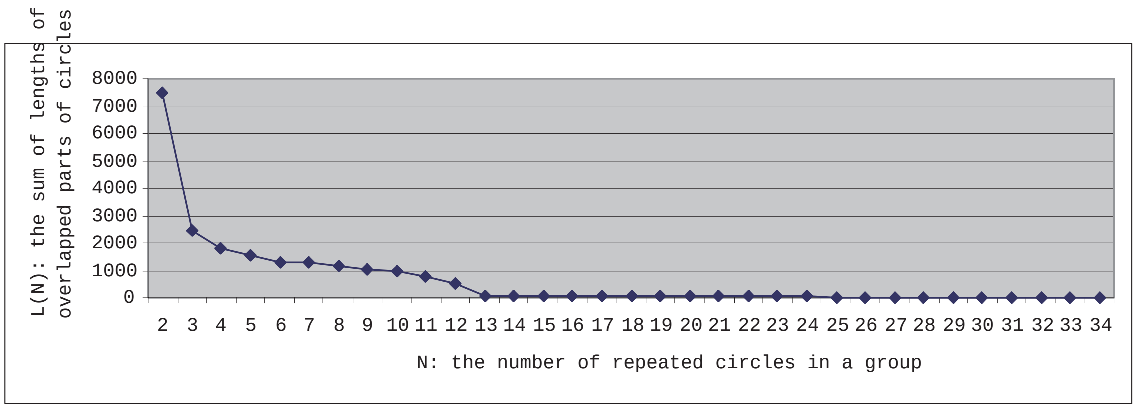


Figure 6: Sum of lengths of overlapping parts of repeated cycles in each group - Travelocity web page [19] in Figure 1

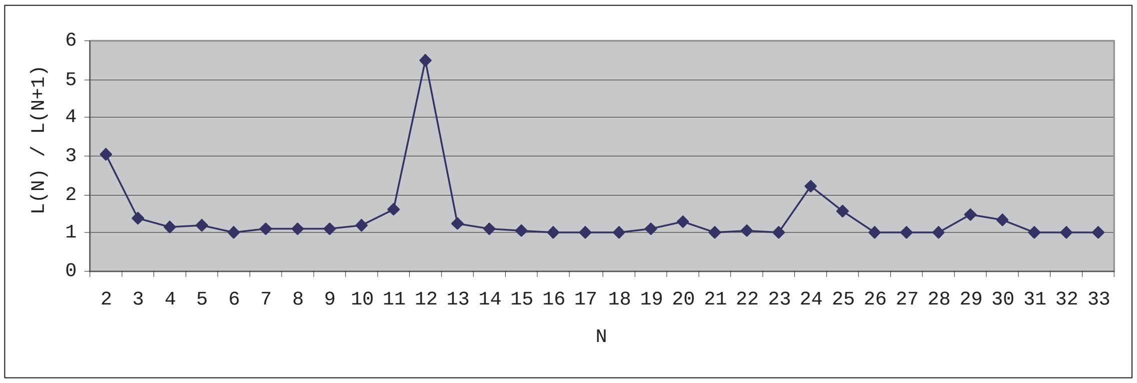


Figure 7: Redraw Figure 6 with ratio: L(N) / L(N+1)

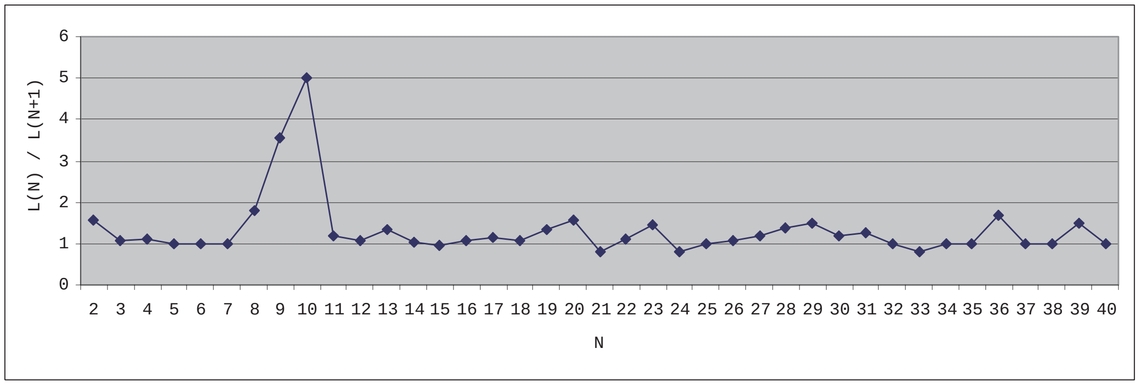


Figure 8: Ratio calculation of the web page at www.whitepapers.com [20]

highest number of tokens.

Assumption 3 *The more tokens a candidate list or table has, the more likely it is to be the correct one.*

On the basis of this assumption, the top 5 groups of cluster sets, ranked by number of tokens they contain, are chosen as the finalists in the process of identifying the number of rows in the list or table.

2.6 Determining the best candidate number of rows

There are two options for identifying the best candidate number of rows. One option is to use a totally unsupervised method that makes the final decision automatically. From observations of tens of web pages, it is concluded that the best group generally has the following features:

- Large number of tokens
- Low dispersion degree
- High ratio $L(N)/L(N+1)$
- High total length of overlapping parts of cycles, i.e., large number of common tokens

Based on these observations, a utility function could be designed to make the final decision on the correct number of rows.

The other option is to involve the user, by asking him/her to select the correct number of rows from among, for example, the top five candidates. Currently we choose this method, which we call "post-supervised" learning, and leave the design of a fully unsupervised approach as future work.

The final group chosen represents the number of rows, and each cycle set in the group corresponds to a candidate table or list. The system further selects the cycle set with the largest number of tokens. The list or table can then be separated into rows according to the cycles in this cycle set.

Data field	**Template part**	**Data field**	**Template part**	**Data field**
A	B	E	D	F
C	B	F	D	A
F	B	A	D	H

Table 2: A simplified example of a table in the page. Each character represents a token sequence. B and D are parts of row template, and other characters are data fields.

3 Inducing Row Template and Identifying Columns

Since all rows in the list and table are properly separated, it is possible to induce a row template from them. The idea behind this is to search for all sequences of tokens that appear in each row.

3.1 Inducing Row Template

In [12], search for a row template starts with the first token on the page and grows a sequence by appending tokens to it, subject to the condition that the sequence appears on every page; then the process continues to create a new sequence from the next unmatched token. Thus a *token sequence* is composed of a set of consecutive tokens on a web page. Each induced token sequence becomes part of the page template. In our experiments, we observed that such an algorithm may fail in some cases. Table 2 is a simplified example of a table with 2 rows. Using the algorithm in [12], only A is recognized as part of the template. The columns are separated as shown in Table 3, which is less likely a table in an actual web page since:

- Data fields in the same column are usually close to each other in length, or number of tokens.
- Data fields in the same column are usually close to each other in the relative positions to their own rows. The details are explained later in this section.

From our experiments, we observe that, to separate the columns correctly, a row template should contain as many token sequences as possible, i.e., the Longest Common Subsequence [4] among rows.

Data field	**Template part**	**Data field**
-	A	BEDF
CBFD	A	-
FB	A	DH

Table 3: Extraction results of Table 2 by Lerman's system

Consider sequences of tokens, $X = \{x(1), x(2), ..., x(m)\}$, $Y = \{y(1), y(2), ..., y(n)\}$, $Z = \{z(1), z(2), ..., z(k)\}$. The following definitions formalize subsequences and related concepts.

Definition 3.1 (Subsequence) *Assuming sequence Z is a subsequence of X if there exists a strictly increasing sequence $\{i_1, i_2, ..., i_k\}$ of indices of X such that for all j = 1, 2, ..., k, there is a $x(i_j) = z(j)$.*

Definition 3.2 (Common Subsequence) *Sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y, and Z is the **Longest Common Subsequence** (LCS) if it is the maximum-length common subsequence of X and Y [4].*

To solve the LCS problem, a dynamic programming algorithm from [4] can be used. Denote by X_j the prefix $\{x(1), x(2), ..., x(j)\}$ of X. If Z is a LCS of X and Y, the following conditions are true:

- If $x(m) = y(n)$ then $z(k) = x(m)$, $\quad Z_{k-1}$ is a LCS of X_{m-1} and Y_{n-1}.
- If $x(m), y(n)$ are different and $z(k)$ is not equal to $x(m)$ then Z is a LCS of X_{m-1} and Y.
- If $x(m)$ and $y(n)$ are different and $z(k)$ is not equal to $y(n)$ then Z if a LCS of X and Y_{n-1}.

A recursive algorithm can be constructed from the above three possibilities and eventually reach a LCS of two data sequences. The algorithm to calculate the LCS of n rows is shown in Figure 9.

```
input: S(n) =  set of data sequences corresponding to n rows
output: LCS = the LCS of n rows
Begin
    LCS = getLCS(S(0), S(1))
    for i= S(2) to S(n-1)
        LCS = getLCS(LCS, S(i))
end
```

Figure 9: The algorithm to calculate the Longest Common Subsequence (LCS) of n rows

3.2 Identifying columns

In order to identify columns, data fields that are not part of the row template need to be extracted. The following features help extract the data fields from the web page:

- Any data field of a column is between two token sequences of the row template.
- Data fields are the actually displayed data on the page.

It is reasonable to assume that most of the data actually displayed consists of everything except HTML tags and control symbols, such as "&nbsp;" and "&#160;". Accordingly, data fields can be extracted by following steps:

1. Pick up all tokens between two consecutive token sequences in all rows and mark as a column.
2. Extract all columns by step 1 and set up as a table.
3. Refine each data field in the table by leaving actually displayed data only.

After identifying columns, the whole list or table in the web page is obtained. For instance, the LCS generated from Example 2 is composed of two token sequences: $< b >$ and $< /b >< /a >< /td >< tdalign = right >$. Data fields separated by the LCS and actually displayed data are listed in table 4.

Data field
AIRPORT HOTEL HALIFAX
INN ON THE LAKE

Table 4: Table extracted from Example 2

4 Matching a Web Page against a Set of Row Templates

Once the row template has been established, it can be used to automatically extract the data fields from the web pages. The number of times the row template is repeated in the page is equal to the number of rows. The problem discussed in this section is how to match a new web page against a set of previously extracted templates. Normally the matching template has the following features:

- Almost all its tokens repeatedly appear in the page with the same order.
- The relative positions of its tokens (representing by token indexes) are close to those of the matched tokens in the page.

Given a new web page, the set of previously extracted row templates can be ranked on the basis of **Similarity** to the web page. Similarity is calculated by calculating the Longest Common Subsequence (LCS)) of the row template and each repeated cycle of tokens of the row template in the web page. The ratio of token numbers of the LCS and the row template is also calculated. Similarity between a row template and a web page is defined as the average of all ratios calculated for the template and the web page. The best matching template is the one with the largest Similarity. The Similarity algorithm is shown in Figure 10.

Based on experiments with the web sites listed in Appendix B, it has been observed that the matching template is usually much more distinguished in both the number and relative position of common tokens than all others. The template with more than 70% tokens matched in the new page will be accepted in our system, since in experiments, most web pages generated at least 70% common tokens with the template that generated them. If no row template has similarity more than 0.7, then no template matches the new page. An

```
input:
    P = the web page
    T = a row template
output:
   Similarity
begin
        C(n) = a set of repeated cycle in P, each of which
        includes tokens of the row template
        Similarity := 0
        for c= C(1) to C(n)
            L = the LCS of c and T
            Similarity := Similarity +
                          numberOfTokens(L) / numberOfTokens(T)
        end for
        Similarity := Similarity / n
end
```

Figure 10: Calculation of Similarity of a web page with a row template

obvious disadvantage of this method is that its computational cost is linear in the number of previously extracted templates. Optimization of the process is a topic for future research.

The repeated cycles generated by the chosen row template may interfere with each other by involving the same part of the page. Some of these are removed based on relative positions of tokens in the page and those in the row template. The relative position can be evaluated by **offset**.

Formula 1 *The offset of a token in a row to a matched token in row template is calculated by:*

$$offset = |(I_n - I_{n0}) - (I_t - I_{t0})| \ / \ L_t \tag{4.3}$$

I_n*: The index in the row of the token, whose offset is to be calculated.*

I_{n0}*: The index in the row of the first token in the row template.*

I_t*: The index in the row template of the token whose offset is to be calculated.*

I_{t0}*: the index of the first token in the row template.*

L_t*: The number of tokens in the row template.*

Example 3.

Consider Example 2 in Section 2.1, and the indices assigned to the tokens. Assume the following is part of a row in the new page:

$... < b > MOTEL\ 6\ PADUCAH < /b >< /td >< td\ align = right\ ...$

The indexes assigned to tokens are:

$70 :< b >$

$71 : MOTEL\ 6\ PADUCAH$

$72 :< /b >$

$73 :< /td >$

$74 :< td\ align = right$

The offset of $< /td >$ in the new page to the matched token in the row template is: $|(73-70)-(4-0)|/6 \approx 0.167$. The offset of a row in the new page to the row template is the sum of all token offsets in the row to the row template. For the above example, the row offset is 0.333.

A template library is generated where all extracted templates, as well as indexes of all tokens for tracking relative token positions, are stored. Given a new web page, the template matching algorithm described above is used to identify a template matching the web page, if available. The overall algorithm works in the following way.

1. For each row template in the template library, its similarity the web page is calculated.

2. The template with the largest similarity (provided it is above 0.7) is chosen and the FSA with repeated cycles corresponding to the chosen template is created.

3. Two cycles may interfere with each other by involving the same part of the page. Hence, the one with the larger offset value is removed.

4. If no template matches the new web page, post-supervised learning is used to identify rows and columns and extract the new template.

Appendix A shows details of the template matching algorithm.

5 Implementation and Performance

5.1 Cooperation with Web robot system

The system described in this paper works in combination with a web robot [15], which obtains web pages including lists or tables. This web automatically queries dynamic web sites on the basis of a script, thus freeing the user from the repetitive form filling and reading of the returned lists and tables associated with activities such as making a car rental or airline flight or hotel reservation. In other words, the web robot system is capable of crawling secure dynamic web sites. Hence, our web robot [15], is able to:

1. Locate target web sites.
2. Establish network communication.
3. Log into the site, if required.
4. Obtain some environmental variables if necessary.
5. Locate the web pages with inquiry forms.
6. Fill and submit the forms.
7. Obtain and store the return web pages for information extraction.

The web robot works as follows:

1. User inputs the address of the web site and links to destination web pages.
2. Robot establishes HTTP or HTTPS (HTTP over Secure Socket Layer) connection.
3. It handles cookies and environmental variables to communicate with server side scripts and log in, if required.
4. It automatically fills and submits the HTML form on the basis of a script containing the required information.
5. It stores response pages from the server for information extraction.

The combination of the information extraction program and the web robot of [15] have been combined as shown in Figure 11. The information extraction system described here receives input from the web robot and outputs the list or table.

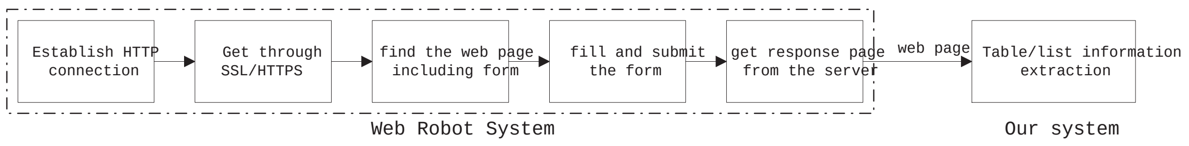

Establish HTTP connection
Get through SSL/HTTPS
find the web page including form
fill and submit the form
get response page from the server
web page
Table/list information extraction
Web Robot System
Our system

Figure 11: Combined system

Hotel name	**Rate**	**Address**	**Distance**
AIRPORT HOTEL HALIFAX	Rate: CAD 95.00 – CAD 124.00*	60 BELL BLVD, Enfield, NS B2T1K3	4.7 km / 2.9 mi
INN ON THE LAKE	Rate: CAD 95.00 – CAD 130.00*	3009 NO 2 HIGHWAY,Waverley, NS B0N2S0	12.6 km / 7.8 mi
COUNTRY DARTMOUTH	Rate: CAD 70.00 – CAD 100.00*	101 YORKSHIRE AVE AT WINDMILL, Dartmouth, NS B2Y3Y2	14.2 km / 8.8 mi
RAMADA PLZ DARTMOUTH HALIFAX	Rate: CAD 99.00 – CAD 220.00*	240 BROWNLOW AVE, Dartmouth, NS B3B1X6	20.3 km / 12.6 mi
BURNSIDE HOTEL	Rate: CAD 55.00 – CAD 129.00*	739 WINDMILL ROAD, Dartmouth, NS B3B1C1	20.5 km / 12.7 mi
BW MIC MAC HOTEL	Rate: CAD 75.00 – CAD 99.99*	313 PRINCE ALBERT ROAD, Dartmouth, NS R2Y1N2	21.6 km / 13.4 mi
FUTURE INNS DARTMOUTH	Rate: CAD 73.00 – CAD 94.98*	20 HIGHFIELD PARK DRIVE, Dartmouth, NS B3A4S8	21.8 km / 13.5 mi
COMFORT INN DARTMOUTH	Rate: CAD 55.00 – CAD 99.00*	456 WINDMILL RD, Dartmouth, NS B3A1J7	22.0 km / 13.7 mi
HOLIDAY INN HALIFAX HARBOURVIEW	Rate: CAD 95.00 – CAD 165.00*	99 WYSE RD, Dartmouth, NS B3A1L9	23.3 km / 14.5 mi
ECONO LODGE HALIFAX	Rate: CAD 64.00 – CAD 98.00*	560 BEDFORD HWY, Halifax, NS B3M2L8	23.3 km / 14.5 mi
MARANOVA SUITES	Rate: CAD 67.50 – CAD 102.00*	65 KING STREET, Dartmouth, NS B2Y4C2	23.3 km / 14.5 mi
EXPRESS HALIFAX	Rate: CAD 89.00 – CAD 149.00*	133 KEARNEY LAKE ROAD, Halifax, NS B3M4P3	24.9 km / 15.4 mi

Table 5: Extraction results of the example in Figure 1

The system has been tested on web pages from the 24 web sites listed in Appendix B. The web sites cover various application areas, such as hotel reservations, book searches, video rentals, looking for driving directions, searching for people and general search engines. Most of lists and tables are extracted with 100 percent accuracy, including irregular ones missing some data fields. Table 5 summarizes the extraction results of the web page shown in Figure 1.

5.2 System performance

Our approach concentrates on learning the row template by identifying common data within single web page. Compared to the page template used in [12], the row template has the following advantages:

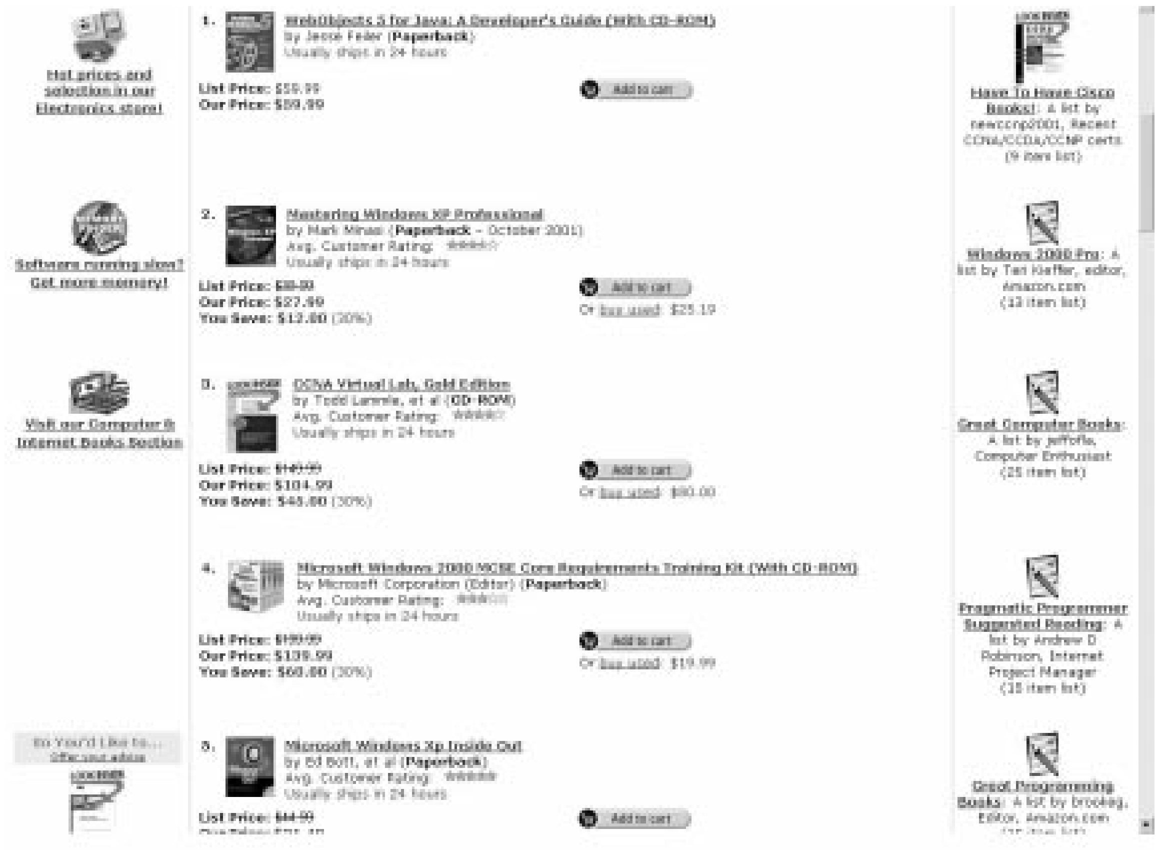
- It can work with a smaller number of pages, even a single page, thus overcoming the potential problem that the number of similar pages available on a particular site at any given time is often quite limited [3].
- Unlike [12], our approach does not require the effort of manually identifying similar web pages as training examples.
- The efficiency is improved by ignoring most of the unrelated web page data outside the lists and tables.
- Rows are identified simultaneously with locating the beginning and end of the row template. This again significantly reduces the complexity of the whole system.
- The row template makes it much easier to identify columns than the page template of [12] by focusing on a single row.

A challenge of adopting the technique of row template instead of page template is that the beginning and end of the row template are much harder to identify because all rows are within one page. This is not required in [12], since the range of the page template is actually the beginning and the end of the page. Thus, an algorithm for quickly searching all possible separations of rows is required in our system. In other words, the implementation of our system is different from reference [12] in that we identify rows first.

We tested our system on the same web sites as in [12], shown in table 6. The accuracy is calculated by the percentage of correctly extracted tuples. Lists or tables in most of examples are correctly extracted. One example, Borders, [18] is not perfectly analyzed, since some data, that do not belong to the list, is present between every 2 rows as shown in Figure 12. In this case, all data fields in the list are successfully extracted but followed by some unnecessary data. We estimate this example as 90 percent correctness. For YahooPeople and Arrow, 5 rows are actually defined in the HTML table, but are displayed as 10 rows visually. Therefore, if the user chooses 5 instead of 10 rows when the system prompts, the results are extracted correctly. We treat these two examples as successfully analyzed. Our system performs with 100% accuracy on the additional 10 sites shown in Appendix B.

Example	**Lerman's system**[12]	**Our system**
Airport	Correct tuples	Correct tuples
Blockbuster	No tuples extracted	correct tuples
Borders	Correct	tuples 90%
cuisineNet	No tuples extracted	Correct tuples
RestaurantRow	Correct tuples	Correct tuples
YahooPeople	Correct tuples	Correct tuples
YahooQuote	18/20 tuples correct	Correct tuples
WhitePapers	Correct tuples	Correct tuples
MapQuest	Tuples begin in the middle of the rows	Correct tuples
Hotel	Correct tuples	Correct tuples
CitySearch	Correct tuples	Correct tuples
CarRental	Correct tuples	Correct tuples
Boston	Correct tuples	Correct tuples
Arrow	No tuples extracted	Correct tuples
Average	**70%**	**99%**

Table 6: Performance of the system of [12] and our system on the 14 examples of [12].



Hot prices and selection in our Electronics store!

Software running slow? Get more memory!

Visit our Computer & Internet Books Section

So You'd Like to... Offer your advice

1. WebObjects 5 for Java: A Developer's Guide (With CD-ROM)
by Jesse Feiler (**Paperback**)
Usually ships in 24 hours
List Price: $59.99
Our Price: $59.99
Add to cart

2. Mastering Windows XP Professional
by Mark Minasi (**Paperback** - October 2001)
Avg. Customer Rating:
Usually ships in 24 hours
List Price: ~~$39.99~~
Our Price: $27.99
You Save: $12.00 (30%)
Add to cart
Or buy used: $25.19

3. CCNA Virtual Lab, Gold Edition
by Todd Lammle, et al (**CD-ROM**)
Avg. Customer Rating:
Usually ships in 24 hours
List Price: ~~$149.99~~
Our Price: $104.99
You Save: $45.00 (30%)
Add to cart
Or buy used: $80.00

4. Microsoft Windows 2000 MCSE Core Requirements Training Kit (With CD-ROM)
by Microsoft Corporation (Editor) (**Paperback**)
Avg. Customer Rating:
Usually ships in 24 hours
List Price: ~~$199.99~~
Our Price: $139.99
You Save: $60.00 (30%)
Add to cart
Or buy used: $19.99

5. Microsoft Windows Xp Inside Out
by Ed Bott, et al (**Paperback**)
Avg. Customer Rating:
Usually ships in 24 hours
List Price: ~~$44.99~~
Add to cart

Have To Have Cisco Books!: A list by newccnp2001, Recent CCNA/CCDA/CCNP certs (9 item list)

Windows 2000 Pro: A list by Teri Kieffer, editor, Amazon.com (13 item list)

Great Computer Books: A list by jeffoflo, Computer Enthusiast (25 item list)

Pragmatic Programmer Suggested Reading: A list by Andrew D Robinson, Internet Project Manager (15 item list)

Great Programming Books: A list by brookeg, Editor, Amazon.com

Figure 12: An example from Borders [18] web site.

IE System	**WIEN**[16]	**STALKER**[16]	**Lerman's**[12]	**Our System**
Learning Type	Supervised	Supervised	Unsupervised	Unsupervised with minimal user interaction
Training Set	labelled	labelled	Unlabelled	Unlabelled
Accuracy	60%	80%	70%	99%
Computer Processing Time	-	-	-	from several seconds to 1 minute
Human Processing Time	-	-	-	choosing one from 5 items

Table 7: Overall performance comparison of WIEN, STALKER, Lerman's and our systems

6 Discussion

A comprehensive performance comparison of the published performance of the systems WIEN, STALKER and Lerman's [12] with our system is shown in Table 7. The accuracies of WIEN and STALKER are given as reported in [16]. However, our system has only been tested on the same web sites as Lerman's [12] since web sites, on which WIEN and STALKER were tested, were not specified in [16].

As shown in Table 7, our system has the highest accuracy, almost 100 percent. For practical applications, accuracy is obviously of critical importance.

The processing time and the manual labelling overhead are two other critical factors. Comparing these Information Extraction systems, WIEN and STALKER need user labeling of all data fields in several rows for each training example, and therefore they require significant human involvement; the system in [12] employs two general unsupervised learning algorithms, AutoClass and DataPro, which are computationally demanding. On the other hand, algorithms in our system are simpler compared to the other systems, thus needing less computer processing time; the implementation of the row template greatly decreases the amount of user involvement compared to WIEN and STALKER. Furthermore, in this work, the user is required to make a simple choice among at most 5 options. Since the user decision is based on visual inspection of a web page, it does not require any particular technical or programming skills or expertise. User involvement is only required the first time the information extraction program is applied to a new site, and whenever there is an update to the web site format.

Our system is designed for all kinds of lists and tables in the web page. Broadly speaking, it is applicable to all data sets with periodical regularity and common features in each period.

We draw the following conclusion about information extraction from our study:

- User labeling of the training data represents the major bottleneck in using wrapper induction techniques [16]. We demonstrate that user labelling of training examples can be avoided while achieving high accuracy.
- Although some user interaction is required in our system, it is minimal compared with fully manual labeling of web pages, as required by the fully supervised information extraction systems.
- Our information extraction system is trainable by non-programmer users, requiring only a basic level of computer usage and ability to visually interpret the web pages of interest.

We plan to continue our work in several directions. First, we will extend the system to work with all the cases described in section 5.2. Second, an order optimization algorithm, as we described in Section 4, may be required to improve the efficiency of the template matching procedure. Third, further research is required to fully automate the determination of the number of rows.

Acknowledgements We thank Dr. M. Heywood for constructive comments, and Jianduan Liang for helpful discussions and technical support. The research was funded by grants from the Natural Sciences and Engineering Research Council of Canada.

References

[1] R. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *Proceedings of the Second International Colloquium on Grammatical Inference and Applications (ICGI94)*, volume 862 of Lecture Notes on Artificial Intelligence, pages 139–152, Berlin, September 1994. Springer Verlag.

[2] P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS): Theory and results. *Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press*, pages 153–180, 1996.

[3] W. Cohen and L. Jensen. A structured wrapper induction system for extracting information from semi-structured documents. In *Automatic Text Extraction and Mining workshop (ATEM-01), IJCAI-01*, Seattle, WA, USA, August 2001.

[4] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Massachusetts, 1989.

[5] M. Degroot and M. Schervish. *Probability and Statistics.* Addison-Wesley Pub. Co., Cambridge, Massachusetts, 1975.

[6] H. Deitel, P. Deitel, and T. Nieto. *Internet and World Wide Web: How to Program.* Prentice-Hall, Upper Saddle River, NJ 07458, 2000.

[7] J. Hanson, R. Stutz, and P. Cheeseman. Bayesian classification theory. Technical report, NASA Ames TR FIA-90-12-7-01, 1991.

[8] M. Hearst. Information integration. *IEEE Intelligent Systems*, Vol. 13, No. 5:12–16, September / October 1998.

[9] C. Knoblock, K. Lerman, S. Minton, and I. Muslea. A machine learning approach to accurately and reliably extracting data from the web. In *IJCAI-2001 Workshop on Text Learning: Beyond Supervision*, Seattle, WA, USA, 2001.

[10] N. Kushmerick. Wrapper induction for information extraction. Technical report, Dept. of Computer Science, U. of Washington, TR UW-CSE-97-11-04, 1997.

[11] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.

[12] K. Lerman, C. Knoblock, and S. Minton. Automatic data extraction from lists and tables in Web sources. In *Automatic Text Extraction and Mining workshop (ATEM-01), IJCAI-01*, Seattle, WA, USA, August 2001.

[13] K. Lerman and S. Minton. Learning the common structure of data. In *In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000), AAAI Press*, pages 609–614, Menlo Park, July 2000.

[14] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, Upper Saddle River, NJ 07458, 1998.

[15] J. Liang, E. Milios, and N. Zincir-Heywood. A robot capable of crawling secure dynamic web sites. Technical report, Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada, 2002.

[16] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.

[17] Tibor G. Rozgonyi. *Statistics for Engineers.* `http://engineering.uow.edu.au/Courses/Stats/File1586.html`, Accessed on Oct. 28, 2002.

[18] `http://www.borders.com/`. *Amazon online shopping site.* Accessed on November 12, 2001.

[19] `http://www.travelocity.com/`. *Travelocity travel site.* Accessed on Oct. 23, 2002.

[20] `http://www.whitepages.com/`. *WhitePages online directory service.* Accessed on Oct. 23, 2002.

A Template Matching Algorithm

The detailed pseudo code of the template matching algorithm is presented below.

```
input:
    P = the new page
    T = a set of template
output:
    C = a set of repeated cycles in FSA
begin
    C = null
    for t = firstTemplate(T) to lastTemplate(T)
        for p = firstToken(P) to lastToken(P)
            totalOffset = 0
            if (p = firstToken(t))
                pos = indexOf(p)
                for p’ = secondToken(t) to lastToken(t)
                    offset= (indexOf(p’) - indexOf(firstToken(t)) - pos
                            + indexOf(p)) / (indexOf(lastToken(t))
                            - indexOf(firstToken(t)))
                    if (p’ = tokenAt(P, pos))
                        totalOffset += |offset|;
                        numOfSame++;
                    else pos++;
                    end if
                end for
                if (numOfSame / numOfToken(t) > 0.7)
                    while (tokenAt(P,pos)!= firstToken(t))
                        pos++
                    end while
                    end = tokenAt(P, pos)
                    append(C, createCycle (p, end, totalOffset))
                end if
            end if
        end for

        // adjust interference of cycles
        i=0;
        while (i< numOfCycle(C))
            cycle1 = cycleAt(C, i)
            cycle2 = cycleAt(C, i+1)
            if (interfered(cycle1, cycle2))
                if (totalOffset(cycle1) > totalOffset(cycle2))
                    remove(C, cycle1)
                else remove(C, cycle2)
                end if
            else i++
            end if
        end while

        if (C corresponds to correct table)
            return C
        end if
    end for
    return null
end
```

B List of Web Sites

The full list of web sites used in our experiments is the following.

- Web sites used in [12].
  - www.airport.com
  - www.blockbuster.com
  - www.borders.com
  - www.cuisinenet.com
  - www.restaurantrow.com
  - www.yahoopeople.com
  - www.yahooquote.com
  - www.whitepapers.com
  - www.mapquest.com
  - www.hotel.com
  - www.citysearch.com
  - www.carrental.com
  - www.boston.com
  - www.arrow.com
- Additional web sites on which our algorithm was tested and had 100% success.
  - www.travelocity.com
  - www.travelnow.com
  - www.hotelreservation.com
  - www.budapesthotels.com
  - www.expedia.com
  - www.khrc.com
  - www.venere.com
  - www.webtourist.com
  - www.google.com
  - www.easyreservation.com