Post-supervised Template Induction for Dynamic Web Sources

Zhongmin Shi, Evangelos Milios, and 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. 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 system, called TIDE (Template Induction for web Data Extraction), 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 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.

However, the World Wide Web has been dominated for a decade by HTML based on a browsing paradigm [1], 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 [2].

A wrapper is a program that enables a Web source to be queried as if it were a database [3]. 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,

Y. Xiang and B. Chaib-draa (Eds.): AI 2003, LNAI 2671, pp. 268–282, 2003.

has been an active area of research in recent years [2, 4]. The first wrapper induction system, WIEN [5] 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 (80%) extraction rules that accept all positive and reject all negative user-labelled training examples [6], but it still requires manually labelled examples.

To overcome the shortcomings of supervised learning, attention is shifting towards unsupervised learning [2], which needs no manually labelled input. In that work, 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.
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 [7], are learned by a pattern learning algorithm.
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 circles that correspond to rows are selected [2]

That system was tested on 14 typical examples[2] with about 70% accuracy. It is undoubtedly a significant effort, but it requires several similar web pages to generate the page template in the first step; the general algorithms used, such as DataPro [7] for learning data pattern and AutoClass for unsupervised classification [8, 9], are computationally intensive, a serious problem in Web applications.

Thus, our work aims to improve the performance of information extraction from lists and tables in the web page. Since high-performance unsupervised learning is rather ambitious, the authors have developed a *Post-supervised* learning technique, called TIDE (Template Induction for web Data Extraction), 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 1 illustrates the whole system at high level.

In the following, identification of rows and columns are described in Sections 2 and 3 respectively. In Section 4 the implementation details of the system and the performance results are given. Finally conclusions are drawn in Section 5.

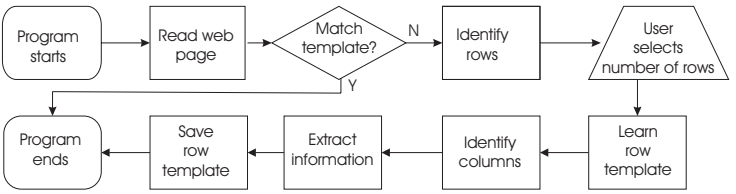

Fig. 1. High level flowchart of the system

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[2]. 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.

Definition 1 (Page template). *A page template is 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.

Definition 2 (Row template). *A row template is 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.

Hence, 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 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 ",.(-)'%", which was selected empirically. The character SPACE is also excluded from the separator list in order to minimize the number of data fields. TABs and NEWLINEs are treated as separators since they are quite likely the separators of the columns and rows.

Definition 4 (Token). *A token is a sequence of characters between separators in the web page. The token includes the separator right after it except for "<", which is included into the token behind it.*

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

Example 1. **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 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.*

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. **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, ô$< b >$ö and ô$< /b >< /a >< /td >< td\ align = right$ö, may be parts of the row template.

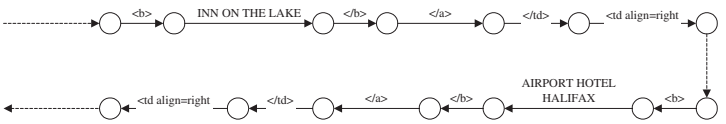

Fig. 2. State diagram of automaton of Example 2

Thus, a grammar induction algorithm [2, 10] 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 $K \times (\Sigma \cup \{e\}) \times K$ [11].

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

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 3 illustrates the automaton of Example 2 after state-merging.

Definition 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 7. ***Length of cycle** is the number of tokens along the cycle.*

Definition 8. ***Cycle set** is a set of cycles that include at least one common state. It corresponds to the candidate list or table.*

Definition 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.*

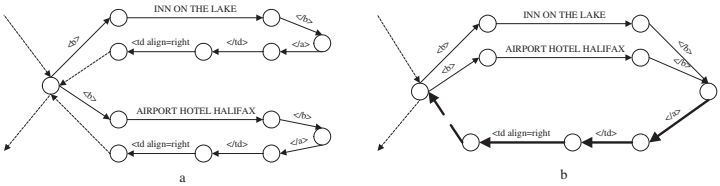

Fig. 3. State diagram of state-merged automaton of Example 2. (a) The automaton with only one pair of states merged. (b) The final automaton after full merging. Thicker lines represent overlapping parts of the cycle set

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.

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 [12] 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, $Disp(S)$ [13] 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.

$$Disp(S) = \sigma(L)/E(L) \tag{1}$$

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.

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 [2]. 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 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, Table 1 shows descriptions of some cycle sets. 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.

Fig. 4. An example of a dynamic web site, www.Travelocity.com[14]. Web clients can submit queries. The Figure illustrates the response page including a list of search results

Table 1. Examples of Cycle sets with N equal to 12 and 13 of the example in Figure 4. 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

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

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.

$$L(N-1)/L(N) << L(N)/L(N+1) \tag{2}$$

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 [14] with 12 rows shown in Figure 4. Figure 5 shows $L(N)$ for groups of cycle sets with varying N. Figure 6 shows the **Ratio** $L(N)/L(N+1)$ as a function of N. The group with 12 repeated cycles clearly stands out. A similar pattern is observed in several other web sites.

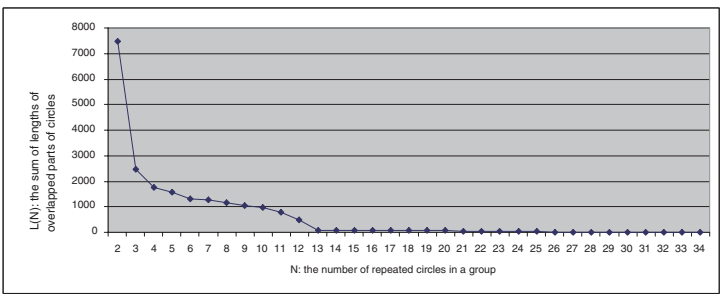

Fig. 5. Sum of lengths of overlapping parts of repeated cycles in each group - Travelocity web page [14] in Figure 4

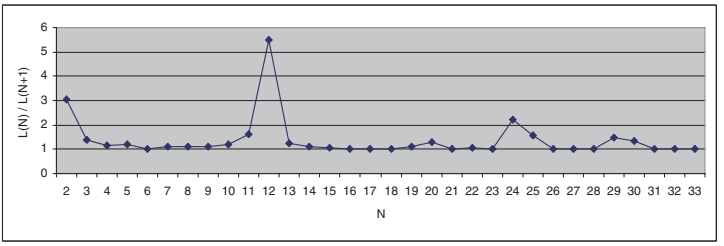

Fig. 6. The ratio L(N) / L(N+1) using the data of Figure 5

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 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. The user is then asked to select the group corresponding to the correct number of rows from among them.

The 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.

3 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. Each induced token sequence becomes part of the page template. Table 2 is a simplified example of a table with 3 rows. From our experiments, to separate the columns correctly, a row template should contain as many token sequences as possible, i.e., the Longest Common Subsequence [15] among rows, 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.

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.

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

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

Definition 10 (Subsequence). *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 11 (Common Subsequence). *Sequence Z is a common subsequence of X and Y if Z is a subsequence of both X and Y. Z is the Longest Common Subsequence (LCS) if it is the maximum-length common subsequence of X and Y [15].*

To solve the LCS problem, a dynamic programming algorithm from [15] 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 $z(k) = x(m)$, 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. Hence, all rows are successfully separated, from which the row template is generated. The next step is to identify 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 " " and " ". 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. Once the row template has been established, it can be used to automatically extract the data fields from the web pages.

4 Implementation and Performance

The system (TIDE) described in this paper works in combination with a web robot [16], which obtains web pages including lists or tables. This web robot 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, and performs the following [16]:

1. Locates target web sites.
2. Establishes network communication.
3. Logs into the site, if required.
4. Obtains environmental variables if necessary.
5. Locates the web pages with inquiry forms.
6. Fills and submits the forms.
7. Obtains and stores the returned 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 information extraction and the web robot of [16] have been combined as shown in Figure 7. The information extraction system described in the previous sections receives input from the web robot and outputs the list or table.

TIDE has been tested on web pages from the 14 web sites of Table 3. 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. Lists and tables are extracted with 100 percent accuracy in 12 out of 14 examples and more than 90 percent accuracy in remaining 2 examples.

Specifically, our approach concentrates on learning the row template by identifying common data within single web page. Compared to the page template used in [2], the row template has the following advantages:

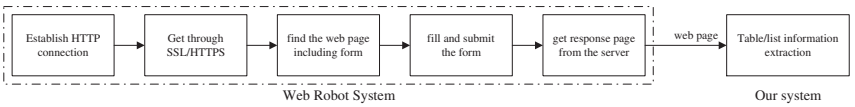

Fig. 7. Combined system

- 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 [17].
- Unlike [2], 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 [2] by focusing on a single row.
- For all lists generated by the same template, once one of them has been analyzed, others can be successfully extracted in a fully unsupervised manner.

We tested our system on the same web sites as in [2], shown in Table 3. 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], visually consists of three columns, in which we are supposed to extract a book list existing in the middle column as shown in Figure 8. Some advertisement data in the left and right columns, however, are physically present between any two rows of the book list in the HTML file. Therefore, all data fields in the list are successfully extracted but followed by some unnecessary data. We estimate this example as 90 percent correctness by data fields in the list in proportion to all data fields actually extracted.

5 Discussion and Conclusions

A comprehensive performance comparison of the published performance of the systems WIEN, STALKER and Lerman's [2] with TIDE is shown in Table 4. The accuracies of WIEN and STALKER are given as reported in [6]. However, TIDE has only been tested on the same web sites as Lerman's [2] since the web sites, on which WIEN and STALKER were tested, were not specified in [6].

As shown in Table 4, our system has the highest accuracy, almost 100 percent. For practical applications, accuracy is obviously of critical importance.

Moreover, the processing time and the manual labelling overhead are two other critical factors. Comparing these Information Extraction systems, WIEN

Table 3. Performance of the system of [2] and TIDE on the 14 examples of [2]. The second to fourth column show the number of rows, columns and lists/tables respectively. The number of rows/columns is that of the most prominent list/table only. In some cases, the number of columns varies depending on whether some data fields could be combined

Example	rows	cols	lists/tables	**Lerman's system**[2]	**TIDE**
Airport	80	4	2	Correct tuples	Correct tuples
Blockbuster	100	4	2	No tuples extracted	Correct tuples
Borders	25	12	5	Correct	Estimated 90%
cuisineNet	8	-	3	No tuples extracted	Correct tuples
RestaurantRow	10	-	3	Correct tuples	Correct tuples
YahooPeople	10	3	1	Correct tuples	Correct tuples
YahooQuote	30	4	1	18/20 tuples correct	Correct tuples
WhitePapers	10	4	2	Correct tuples	Correct tuples
MapQuest	16	-	2	Tuples begin in the middle of the rows	Correct tuples
Hotel	25	5	1	Correct tuples	Correct tuples
CitySearch	20	-	2	Correct tuples	Correct tuples
CarRental	16	-	3	Correct tuples	Correct tuples
Boston	20	4	3	Correct tuples	Correct tuples
Arrow	10	3	1	No tuples extracted	Correct tuples
Average				**70%**	**99%**

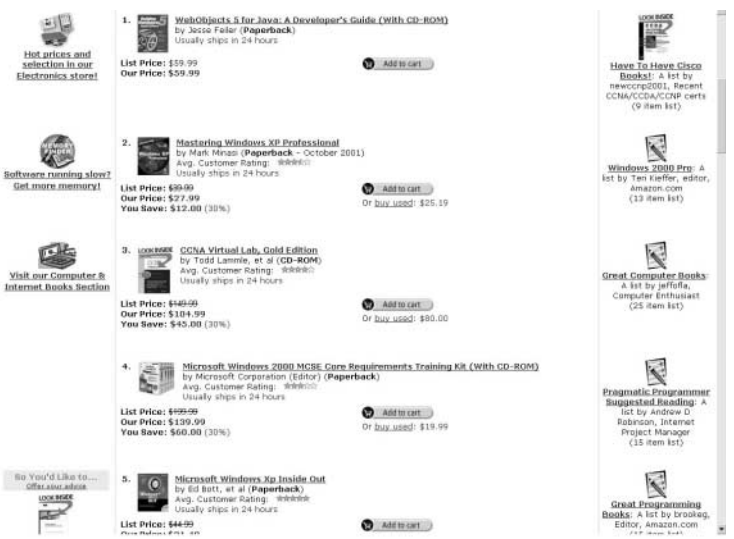

Fig. 8. An example from Borders [18] web site

Table 4. Overall performance comparison of WIEN, STALKER, Lerman's and TIDE

IE System	**WIEN**[6]	**STALKER**[6]	**Lerman's**[2]	**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

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 [2] 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; 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. The design of a totally unsupervised approach is left for future research.

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.

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] Deitel, H., Deitel, P., Nieto, T.: Internet and World Wide Web: How to Program. Prentice-Hall, Upper Saddle River, NJ 07458 (2000) 268

[2] Lerman, K., Knoblock, C., Minton, S.: Automatic data extraction from lists and tables in Web sources. In: Automatic Text Extraction and Mining workshop (ATEM-01), IJCAI-01, Seattle, WA, USA (2001) 268, 269, 270, 272, 273, 278, 279, 280, 281

[3] Knoblock, C., Lerman, K., Minton, S., Muslea, I.: 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) 268
[4] Kushmerick, N.: Wrapper induction: Efficiency and expressiveness. Artificial Intelligence **118** (2000) 15–68 269
[5] Kushmerick, N.: Wrapper induction for information extraction. Technical report, Dept. of Computer Science, U. of Washington, TR UW-CSE-97-11-04 (1997) 269
[6] Muslea, I., Minton, S., Knoblock., C.: Hierarchical wrapper induction for semistructured information sources. Journal of Autonomous Agents and Multi-Agent Systems **4** (2001) 93–114 269, 279, 281
[7] Lerman, K., Minton, S.: Learning the common structure of data. In: In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000), AAAI Press, Menlo Park (2000) 609–614 269
[8] Cheeseman, P., Stutz, J.: Bayesian classification (AUTOCLASS): Theory and results. Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press (1996) 153–180 269
[9] Hanson, J., Stutz, R., Cheeseman, P.: Bayesian classification theory. Technical report, NASA Ames TR FIA-90-12-7-01 (1991) 269
[10] Carrasco, R., Oncina, J.: 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., Berlin, Springer Verlag (1994) 139–152 272
[11] Lewis, H., Papadimitriou, C.: Elements of the Theory of Computation. Prentice-Hall, Upper Saddle River, NJ 07458 (1998) 272
[12] Degroot, M., Schervish, M.: Probability and Statistics. Addison-Wesley Pub. Co., Cambridge, Massachusetts (1975) 273
[13] Rozgonyi, T. G.: Statistics for Engineers. http://engineering.uow.edu.au/Courses/Stats/File1586.html, (Accessed on Oct. 28, 2002) 273
[14] http://www.travelocity.com/: Travelocity travel site. (Accessed on Oct. 23, 2002) 274, 275
[15] Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms. MIT Press, Cambridge, Massachusetts (1989) 276, 277
[16] Liang, J., Milios, E., Zincir-Heywood, N.: A robot capable of crawling secure dynamic web sites. Technical report, Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada (2002) 278
[17] Cohen, W., Jensen, L.: 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 (2001) 279
[18] http://www.borders.com/: Amazon online shopping site. (Accessed on November 12, 2001) 279, 280