

Mining Travel Resources on the Web Using L-Wrappers

Elvira Popescu¹, Amelia Bădică², and Costin Bădică¹

¹ University of Craiova, Software Engineering Department
Bvd.Decebal 107, Craiova, RO-200440, Romania
{badica.costin, popescu.elvira}@software.ucv.ro

² University of Craiova, Business Information Systems Department
A.I.Cuza 13, Craiova, RO-200585, Romania
ameliabd@yahoo.com

Abstract. The work described here is part of an ongoing research on the application of general-purpose inductive logic programming, logic representation of wrappers (L-wrappers) and XML technologies (including the XSLT transformation language) to information extraction from the Web. The L-wrappers methodology is based on a sound theoretical approach and has already proved its efficacy on a smaller scale, in the area of collecting product information. This paper proposes the use of L-wrappers for tuple extraction from HTML in the domain of e-tourism. It also describes a method for translating L-wrappers into XSLT and illustrates it with the example of a real-world travel agency Web site.

1 Introduction

E-tourism is a leading area in e-commerce, with an increasing number of travel agencies offering their services through online transaction brokers ([12]). They provide to human users information in areas like hotels, flights, trains or restaurants, in order to help them to plan their business or holiday trips. The travel information is heterogeneous and distributed, and there is a need to gather, search, integrate and filter it efficiently ([8]). Typically, this information is published by dynamically filling-in HTML templates with structured data taken from relational databases. Therefore, collecting information about travel resources and converting it to a form suitable for automated processing is an appropriate task for information extraction using machine learning ([10]).

This paper approaches the problem of mining travel resources using logic wrappers (L-wrappers) ([1]). As shown in [2], L-wrapper technology was successfully applied to extract tuples from Web pages written in HTML.

Note that the application of logic representations and machine learning to information extraction from the Web is not an entirely new field; several approaches and tool descriptions have already been proposed and published ([5–7, 10, 11, 13, 16, 18]). In our opinion, the advantage of our proposal is the use of the right tool for tackling a given task, i.e. for learning extraction rules it employs inductive logic programming (ILP) systems ([15]), and for performing the extraction it employs XSLT technology ([4]).

The rest of the paper is structured as follows. First, we give a short overview of L-wrapper theory. In section 3 we discuss an approach of translating L-wrappers into XSLT stylesheets. We outline a translation algorithm that is then demonstrated with the

help of an example. In section 4 we present some experiments and discuss their results. Finally, we conclude the paper, outlining some future research directions.

2 L-Wrappers Background

Web pages (including those publishing travel information) can be regarded as semi-structured data modeled as labeled ordered trees. A wrapper takes a labeled ordered tree and returns a subset of tuples of extracted nodes. L-wrappers use patterns defined as logic rules, which can be learned by using general-purpose ILP systems.

For our purposes, it is convenient to abstract labeled ordered trees as sets of nodes on which certain relations and functions are defined ([1]). Basically, we define two relations between tree nodes: the "parent-child" relation and the "next-sibling" linear ordering relation on the set of children of a node. Furthermore, a label is attached to each tree node, modeling a specific tag from a finite set of tag symbols Σ .

In this framework, a pattern is a labeled directed graph with arc labels specifying parent-child and next-sibling relationships and vertex labels specifying conditions such as first child ('f'), last child ('l'), a tag from Σ , or a combination thereof. A subset of the graph vertices is used for selecting the items for extraction (there is an extraction vertex for each attribute name in the relational representation of the information source).

According to the model-theoretic semantics of patterns ([1]), a labeled ordered tree is an interpretation domain for the patterns. Intuitively, patterns are matched against parts of a target labeled ordered tree. In order to have a successful matching, the labels of pattern vertices and arcs must be consistent with the corresponding relations and functions defined over tree nodes. The result of applying a pattern to a semi-structured information source is a set of extracted tuples. An extracted tuple is modeled as a function from attribute names to tree nodes, as in standard relational data modeling.

We propose a semi-automatic L-wrapper development process using ILP ([2]). During this process, two useful operations on patterns were defined ([1]): i) *pattern simplification* – the process of removing arcs in the pattern directed graph without changing the pattern semantics; more precisely we can shift one position right an arc labeled with 'c' in a pattern and we obtain an equivalent pattern. This operation is used to normalize patterns (i.e. bring patterns in a form in which the out-degree of every pattern vertex is at most 1); ii) *pattern merging* – the process of combining simpler patterns that share attributes into more complex patterns, that are capable to extract tuples of higher arity.

An L-wrapper can be defined as a set of patterns that share the set of attributes from the relation scheme of the information source. In this paper we restrict our attention to single-pattern L-wrappers that can be concisely defined in two steps: i) define the pattern graph together with arc labels that model parent-child and next-sibling relations and ii) extend this definition with vertex labels that model conditions on vertices, extraction vertices and assignment of extraction vertices to attributes.

Definition 1. (*Pattern graph*) Let \mathcal{W} be a set denoting all vertices. A pattern graph G is a quadruple $\langle A, V, L, \lambda_a \rangle$ such that $V \subseteq \mathcal{W}$, $A \subseteq V \times V$, $L \subseteq V$ and $\lambda_a : A \rightarrow \{ 'c', 'n' \}$. The set \mathcal{G} of pattern graphs is defined inductively as follows:

- i) If $v \in \mathcal{W}$ then $\langle \emptyset, \{v\}, \{v\}, \emptyset \rangle \in \mathcal{G}$

ii) If $G = \langle A, V, L, \lambda_a \rangle \in \mathcal{G}$, $v \in L$, and $w, u_i \in \mathcal{W} \setminus V$, $1 \leq i \leq n$ then a) $G_1 = \langle A \cup \{(w, v)\}, V \cup \{w\}, (L \setminus \{v\}) \cup \{w\}, \lambda_a \cup \{(w, v), 'n'\}\rangle \in \mathcal{G}$; b) $G_2 = \langle A \cup \{(u_1, v), \dots, (u_n, v)\}, V \cup \{u_1, \dots, u_n\}, (L \setminus \{v\}) \cup \{u_1, \dots, u_n\}, \lambda_a \cup \{(u_1, v), 'c'\}, \dots, (u_n, v), 'c'\}\rangle \in \mathcal{G}$; c) $G_3 = \langle A \cup \{(w, v), (u_1, v), \dots, (u_n, v)\}, V \cup \{w, u_1, \dots, u_n\}, (L \setminus \{v\}) \cup \{w, u_1, \dots, u_n\}, \lambda_a \cup \{(w, v), 'n'\}, (u_1, v), 'c'\}, \dots, (u_n, v), 'c'\}\rangle \in \mathcal{G}$;

Intuitively, if $\langle A, V, L, \lambda_a \rangle$ is a pattern graph then V are its vertices, A are its arcs, $L \subseteq V$ are its leaves (vertices with in-degree 0) and λ_a indicates parent-child and next-sibling arcs. Examples can be found in [1].

Definition 2. (Single-pattern L-wrapper) Let \mathcal{A} be the set of attribute names. A single-pattern L-wrapper is a tuple $W = \langle V, A, U, D, \mu, \lambda_a, \lambda_c \rangle$ such that $\langle A, V, L, \lambda_a \rangle$ is a pattern graph, $U = \{u_1, u_2, \dots, u_k\} \subseteq V$ is the set of pattern extraction vertices, $D \subseteq \mathcal{A}$ is the set of attribute names, $\mu : D \rightarrow U$ is a one-to-one function that assigns a pattern extraction vertex to each attribute name, and $\lambda_c : V \rightarrow C$ is the labeling function for vertices. $C = \{\emptyset, \{f'\}, \{l'\}, \{\sigma\}, \{f', l'\}, \{f', \sigma\}, \{l', \sigma\}, \{f', l', \sigma\}\}$ is the set of conditions, where σ is a label in the set Σ of tag symbols.

3 Translating L-Wrappers into XSLT Stylesheets

The process of information extraction from the Web can be structured into a sequence of stages ([1, 2]): page collection, pre-processing, manual information extraction, conversion to the input format of the learning program, learning, wrapper compilation, wrapper execution. This section addresses wrapper compilation, namely the translation of L-wrappers into XSLT. Actually, we will use a subset of XSLT, called XSLT₀ ([3]).

The output of the learning stage is a set of rules. Rules are first converted into graph-like descriptions as introduced by definitions 1 and 2. These graphs are further processed using pattern operations to produce the final graph description of the wrapper.

At this point we apply an algorithm for translating L-Wrappers into XSLT that exploits the graph-like definition of L-wrappers. The idea is to generate XSLT₀ templates for all the leaf and extraction vertices of the L-wrapper, moving upwards and downwards in the graph. The extracted information is passed between templates by means of template variables and parameters. Here is an informal description of this algorithm:

Step 1. Start from the document root and generate the start template, by moving downwards to one of the vertices in $L \cup U$.

Step 2. Move from the current vertex (say w_0) to another vertex in $L \cup U$ (say w_1). The path taken depends on the type of the first vertex: if $w_0 \in L$ then we move first upwards, to the common ascendent of w_0 and w_1 and then downwards to w_1 ; if $w_0 \notin L$ then we follow the direct descendent path to w_1 .

Step 3. Generate a template that will select the content of w_0 in case $w_0 \in U$.

Step 4. Repeat steps 2 and 3 until there are no more unvisited vertices in $L \cup U$.

Step 5. Generate the final template, which will display the extracted tuples.

4 Experiments

We now demonstrate the use of L-wrappers to extract travel information from the Travelocity Web site (see [17] and figure 1). That Web page displays hotel information

comprising the hotel name, address and description, the check-in and check-out dates, the types of rooms offered and the corresponding average nightly rate. Adopting the relational model, we associate to this resource the following set of attribute names related to hotels: $\{name, address, description, period, roomtype, price\}$.

Gresham Belson Hotel				Lowest Avg Nightly Rate: \$216.54
★★★★	Map this Property	Write a Review of this Hotel	Low Rates GUARANTEED	
CHAUSSEE DE LOUVAIN 805 Brussels, BE 1140 The Gresham Belson Hotel is situated approximately four miles from the city center and five miles from Brussels International Airport.				
Jun 1 thru Jun 3				
Room Type	Wed	Thu	Avg Nightly Rate*	
Standard Room	\$216.54	\$216.54	\$216.54	Select
Crowne Plaza Brussels Airport				Lowest Avg Nightly Rate: \$211.92
★★★★	Map this Property	Write a Review of this Hotel	Low Rates GUARANTEED	1 Traveler Review
DA VINCILAAN 4 Driegem, BE 1831 The Crowne Plaza Brussels Airport is situated less than two miles from Brussels International Airport and nine miles from downtown Driegem.				
Jun 1 thru Jun 3				
Room Type	Wed	Thu	Avg Nightly Rate*	
Standard Room	\$211.92	\$211.92	\$211.92	Select
Deluxe Room	\$267.15	\$267.15	\$267.15	Select

Fig. 1. An XHTML document fragment and its graphic view

Because of the relatively large number of attributes, we used the pattern merging approach. This overcomes the difficulty of directly learning tuples with arity greater than 2 ([2]). The following pairs of attributes were chosen: $\{name, address\}$, $\{address, description\}$, $\{name, period\}$, $\{period, roomtype\}$, and $\{roomtype, price\}$.

Then we generated extraction rules for each pair of attributes by using the FOIL program ([15]), as described in [2]. The following 5 rules were generated: ($NA = name$, $AD = address$, $DE = description$, $PE = period$, $RO = roomtype$, and $PR = price$):

```

extract(NA, AD) ← first(AD) ∧ td(AD) ∧ child(C, NA) ∧ child(D, AD) ∧ next(D, E) ∧ child(F, E) ∧
span(C) ∧ first(D) ∧ child(G, C) ∧ child(H, G) ∧ next(I, H) ∧ child(J, I) ∧ child(K, J) ∧ child(L, K) ∧
child(M, J) ∧ child(N, M) ∧ child(O, L) ∧ child(P, O) ∧ child(Q, P) ∧ child(N, Q).
extract(AD, DE) ← child(C, AD) ∧ child(D, DE) ∧ next(AD, E) ∧ next(C, F) ∧ child(G, F) ∧ child(F, D) ∧
first(G) ∧ text(DE).
extract(NA, PE) ← text(NA) ∧ child(C, NA) ∧ child(D, PE) ∧ next(E, D) ∧ child(F, E) ∧ b(D) ∧
child(G, C) ∧ child(H, G) ∧ next(I, H) ∧ child(J, I) ∧ child(K, J) ∧ next(K, L) ∧ next(L, M) ∧ child(M, N) ∧
child(O, F) ∧ child(P, O) ∧ child(N, P).
extract(PE, RO) ← child(C, PE) ∧ child(D, RO) ∧ next(D, E) ∧ next(F, C) ∧ child(G, E) ∧ child(H, F) ∧
next(I, G) ∧ child(J, I) ∧ next(K, J) ∧ first(D) ∧ child(K, L) ∧ child(L, H).
extract(RO, PR) ← child(C, RO) ∧ child(D, PR) ∧ next(C, E) ∧ next(D, F) ∧ child(G, E) ∧ next(H, G) ∧
child(I, H) ∧ next(J, I) ∧ child(G, D) ∧ first(C) ∧ last(F) ∧ text(PR).

```

Next we merged these patterns into a single-pattern L-wrapper and then we translated it into XSLT₀ using the algorithm outlined in section 3. A set of 7 XSLT₀ templates was obtained (see table 1).

theoretical work, we are interested in giving a formal proof of the correctness of the mapping of L-wrappers to XSLT.

References

1. Bădică, C., Bădică, A.: Logic Wrappers and XSLT Transformations for Tuples Extraction from HTML. In: Bressan, S.; Ceri, S.; Hunt, E.; Ives, Z.G.; Bellahsene, Z.; Rys, M.; Unland, R. (eds): *Proc. 3rd International XML Database Symposium XSym'05*, Trondheim, Norway. LNCS 3671, Springer-Verlag (2005), 177–191
2. Bădică, C., Bădică, A., Popescu, E.: Tuples Extraction from HTML Using Logic Wrappers and Inductive Logic Programming. In: Szczepaniak, P.S., Kacprzyk, J., Niewiadomski, A. (eds.): *Proc.AWIC'05*, Lodz, Poland. LNAI 3528 Springer-Verlag (2005), 44–50
3. Bex, G.J., Maneth, S., Neven, F.: A formal model for an expressive fragment of XSLT. *Information Systems*, No.27, Elsevier Science (2002), 21–39.
4. Clark, J.: XSLT Transformation (XSLT) Version 1.0, W3C Recommendation, 16 November 1999, <http://www.w3.org/TR/xslt> (1999).
5. Chidlovskii, B.: Information Extraction from Tree Documents by Learning Subtree Delimiters. In: *Proc. IWeb'03*, Acapulco, Mexico (2003), 3–8
6. Freitag, D.: Information extraction from HTML: application of a general machine learning approach. In: *Proc. AAAI'98*, (1998), 517–523
7. Ikeda, D., Yamada, Y., Hirokawa, S.: Expressive Power of Tree and String Based Wrappers. In: *Proc. IWeb'03*, Acapulco, Mexico, (2003), 16–21.
8. Knoblock, C.: Agents for Gathering, Integrating, and Monitoring Information for Travel Planning. In: *Intelligent Systems for Tourism. IEEE Intelligent Systems*. Nov./Dec. (2002), 53–66.
9. Kosala, R., Bussche, J. van den, Bruynooghe, M., Blockeel, H.: Information Extraction in Structured Documents Using Tree Automata Induction. In: *Principles of Data Mining and Knowledge Discovery, 6th European Conf.*, Helsinki, Finland, LNAI 2431, Springer-Verlag (2002), 299–310.
10. Kushmerick, N., Thomas, B.: Adaptive Information Extraction: Core Technologies for Information Agents, In: *Intelligent Information Agents R&D in Europe: An AgentLink perspective* (Klusch, Bergamaschi, Edwards & Petta, eds.). LNAI 2586, Springer-Verlag (2003), 79–103.
11. Laender, A.H.F., Ribeiro-Neto, B., Silva, A.S., Teixeira, J.S.: A Brief Survey of Web Data Extraction Tools. In: *SIGMOD Record*, Vol.31, No.2, ACM Press (2002), 84–93.
12. Laudon, K.C., Traver, C.G.: *E-commerce. business. technology. society* (2nd ed.). Pearson Addison-Wesley, (2004).
13. Li, Z., Ng, W.K.: WDEE: Web Data Extraction by Example. In: L. Zhou, B.C. Ooi, and X. Meng (Eds.): *Proc.DASFAA'2005*, Beijing, China. LNCS 3453, Springer-Verlag (2005), 347–358.
14. Oxygen XML Editor. <http://www.oxygenxml.com/>.
15. Quinlan, J. R., Cameron-Jones, R. M.: Induction of Logic Programs: FOIL and Related Systems, *New Generation Computing*, 13, (1995), 287–312.
16. Sakamoto, H., Arimura, H., Arikawa, S.: Knowledge Discovery from Semistructured Texts. In: Arikawa, S., Shinohara, A. (eds.): *Progress in Discovery Science*. LNCS 2281, Springer-Verlag (2002), 586–599.
17. Travelocity Web site. <http://www.w3.org/TR/xslt>.
18. Xiao, L., Wissmann, D., Brown, M., Jablonski, S.: Information Extraction from HTML: Combining XML and Standard Techniques from IE from the Web. In: Monostori, L., Vancza, J., Ali, M. (eds.): *Proc. IEA/AIE 2001*. LNAI 2070, Springer-Verlag (2001), 165–174.