Web Wrapper Induction: A Brief Survey

Sergio Flesca [a,*], Giuseppe Manco [b], Elio Masciari [b], Eugenio Rende [a], and Andrea Tagarelli [a]

[a] *DEIS - Univ. of Calabria, 87030 Rende - Italy*
E-mail: {flesca,erende,tagarelli}@si.deis.unical.it
[b] *ICAR-CNR, 87030 Rende - Italy*
E-mail: {manco,masciari}@icar.cnr.it

Nowadays several companies use the information available on the Web for a number of purposes. However, since most of this information is only available as HTML documents, several techniques that allow information from the Web to be automatically extracted have recently been defined. In this paper we review the main techniques and tools for extracting information available on the Web, devising a taxonomy of existing systems. In particular we emphasize the advantages and drawbacks of the techniques analyzed from a user point of view.

Keywords: Information Extraction, Wrapper generation.

1. Introduction

Nowadays several companies use the information available on the Web for several purposes ranging from competitor analysis to automatic collection of news. Indeed several systems have been designed in order to effectively retrieve interesting information. Typically, these systems browse and query the Web to detect the desired information. However, since most of this information is only available as HTML documents, the collected information is not suitable for insertion in the information systems of the companies. Indeed considerable human effort is needed to convert the collected information into a format that can be profitably handled by automatized systems. In order to manage Web data effectively, a user needs to extract relevant information, understand its semantic structure, and convert it into the right format for insertion in the target information system.

Information extraction (IE) techniques aim at transforming unstructured textual information into structured information by means of extraction rules which identify relevant information. Extraction rules are used both to recognize the portions of a document containing relevant data, and to assign a semantic meaning to the identified information, in order to translate it into a structured format. A set of extraction rules suitable to extract information from a Web site is called a *wrapper*.

The two main issues concerning information extraction systems are the development of powerful languages for expressing extraction rules and the capability of generating these rules with the lowest human effort. The latter issue leads to designing several systems for *wrapper generation*.

Two main approaches for the design of wrapper generation tools have been proposed during the last years: the *knowledge engineering* and *automatic training* approach. In the former approach, the extraction rules are designed by a domain expert, according to his/her background knowledge of document characteristics. Clearly, in such an approach the user skills play a crucial role in the successful identification and extraction of relevant information. The automatic training approach instead exploits AI techniques to induce extraction rules starting from a set of information patterns that are marked for extraction by a user.

In the remainder of this paper, we first discuss the main features of both the approaches from the user's point of view and then analyze in more detail the properties of the currently available systems following the automatic training approach.

2. Wrapper Generation

Traditional IE tools extract relevant information from textual documents on the basis of combinations of syntactic and semantic constraints. On the other hand, wrapper generation systems deal with

[*]S. Flesca and E. Rende are partially supported by the EU project Infomix and by Lixto Software GmbH.

Web pages containing heterogeneous mixtures of information and structure. Indeed, the structure of the information contained in a Web document is defined by means of formatting markup tags. Wrapper generation systems can exploit these formatting instructions to define suitable wrappers for a set of similarly structured Web pages. However, wrappers based on very simple extraction rules are not able to effectively exploit these formatting instructions when dealing with pages with a complex structure.

A first rough classification of Web wrappers can be devised by considering the kind of pages that each wrapper is able to deal with. We consider three different types of Web pages:

– *unstructured pages*: some linguistic skills are required to learn relevant features from the pages;
– *semi-structured pages*: such pages do not conform to a fixed schema, in the sense that there is no separate description for the type of data;
– *structured pages*: the information about the structure is available, thus information can only be extracted according to syntactic constraints.

However, not only the structure of a Web page can be used by wrappers: the hyperlink structures of a Web site can also be exploited to extract information. Indeed, hyperlinks are particularly relevant when the needed information is nested in different pages.

As an example consider a set of pages obtained by performing a query on an on-line catalog of high-tech products. Depending on the type of Web search engine we can have the following kinds of results:

– *one-level one-page* result (i.e. one page containing all the item descriptions),
– *one-level multi-pages* (i.e. a chain of pages such that each page contains item descriptions),
– *two-level pages* (i.e. a chain of pages that contains short item descriptions; each short description points to a detailed page on the item).

The main features that wrapper generation systems have to deal with are *scalability* and *flexibility* for the generated wrappers. The former is important for handling a large number of Web sites, the latter is needed to avoid frequent rebuilding of wrappers due to changes occurring in the structure of the wrapped pages.

Wrapper generation can be accomplished in three different ways: manually, semi-automatically and automatically. For manually generated wrappers a deep knowledge of the structure of the documents being wrapped and the coding of suitable parsers for such a structure is required. To speed up this process, in the last five years great effort has been devoted to generating template modules which can be semi-automatically adapted to different sources. Automatic techniques for wrapper generation usually make use of supervised machine learning techniques. In these systems users are required to label relevant data from a set of Web pages that will be used as a training set for the learning process. The features of a training set can be exploited by a system to extract rules.

Obviously, hand-written wrappers perform better than automatically generated ones. The main drawback of hand-written wrappers is that manual development can be very tedious and sometimes too difficult to accomplish.

An interesting taxonomy of wrapper generation tools could be provided based on the involved AI techniques [13,7]:

– Wrapper generation tools based on *languages for wrapper generation* (such as, TSIMMIS [10], Minerva [5] and Web-OQL [2]) which mainly aim at supporting users in constructing wrappers.
– *HTML aware* tools (such as, Lixto [3] and RoadRunner [6]) which are wrapper generation tools exploiting suitable representations of the original HTML documents.
– Wrapper generation tools (such as, RAPIER [4], SRV [9], WHISK [16]) which deal with Web pages containing mostly free text (e.g. rental and job advertisements, announcements). Such tools are based on natural language processing (NLP) techniques such as filtering, syntactic and semantic tagging.
– *Wrapper induction tools* which work relaxing linguistic constraints and paying more attention to formatting features. Examples of wrapper generation tools following such a strategy are Wien [12], SoftMealy [11] and STALKER [14].
– *Modelling-based tools* which try to identify document objects matching a predefined structure [1].

– Finally, *ontology-based tools* which exploit preexisting ontologies and extract information relying directly on data [8].

2.1. Wrapper generation from the user viewpoint

The classification provided in the previous section is not completely satisfactory since it does not take into account the user effort needed to design a wrapper, and moreover it does not consider the category of Web pages to be wrapped. In this section, we give an alternative taxonomy for wrapper induction systems, trying to group the systems according to the key features that allow *users* to build good wrappers.

In particular, a characteristic that needs to be considered is the *structure* of the pages being wrapped. We can distinguish two main categories of systems: 1) systems that are tailored for structured pages and 2) systems designed for pages with a less severe structure. Systems such as ShopBot, Wien, SoftMealy and STALKER are classified into the first group while the second group includes systems such as RAPIER, SRV and WHISK.

Another relevant feature for users is the *degree of automation* of the wrapper generation system, i.e. the effort needed to design a wrapper. Obviously, if a language for wrapper generation is used, the user interaction is very high: the user is required to analyze the document source code, to search for interesting features, and to write some suitable code. A wrapper induction system is used here to avoid hand-writing code since the process is automatic or almost semi-automatic. Examples of these systems are RoadRunner (a fully automatic tool) and BYU [8] (that requires a preliminary construction of an ontology by a domain expert).

Another relevant characteristic is the usability of the tools considered. Many tools provide a graphical user interface: HTML-aware, NLP based, and wrapper induction tools are well-suited for this purpose. Some systems that offer visual wizards for designing wrappers are W4F [15] and Lixto.

As a final key feature, we mention the *language* used for pattern presentation. For example, extraction patterns in WHISK are regular expressions describing *context* and *delimiters* of the phrases to be extracted. SRV generates first-order-logic extraction patterns using some suitable tests on the attribute values and the relational structure of documents.

3. Wrapper induction systems

In this section we describe some wrapper induction systems. As mentioned above, when considering wrapper schemes examples are needed to achieve at the best wrapper definition. Thus we can distinguish between systems that need labelled examples (such as SRV, STALKER,WHISK and WIEN) and systems that can search for important information (such as, ShopBot and Wien) autonomously. The main characteristics of the cited wrapper induction systems are described in the following. In particular, the following description takes into account the overall structure of a system, the kind of documents a system can deal with, and the robustness of the wrappers generated. In such a context, robustness means the capability of a wrapper to deal with small variations within the document, such as missing items or permutations in the order of the items to be extracted. A further aspect that is worth considering is the structuring level of the extracted information. Obviously, a wrapper system capable of only separating sentences within a document is less expressive than a system which is able to extract complex structures (e.g. lists of records) from it.

ShopBot is an agent devoted to extracting information from pages related to Web services (e.g. e-commerce sites). The system works in two main stages by combining heuristic, pattern matching and inductive learning techniques. In the first phase, ShopBot analyzes the target Web site in order to learn its structure and to find the pages containing relevant information. In the second phase, some heuristics are exploited to produce a suitable description of the selected pages. Once a set of descriptions is generated, the system performs a ranking process to identify the best description available. A simple ranking function takes into account the number of occurrences of a given item description. A limitation of ShopBot is that it is unable to label extracted features, so that a manual labelling process has to be carried out.

WIEN (Wrapper Induction ENvironment) is the oldest tool for inductive wrapper generation. WIEN operates on structured texts containing information organized in a tabular fashion. In order to speed-up the learning process, the system automatically labels the training pages exploiting various domain-specific heuristics. The wrapper generation process uses a bottom-up inductive learning technique. More precisely, the algorithm searches the space of all possible delimiters for a wrapper compatible with the training set. The system only deals with documents exhibiting a fixed structure (e.g. it does not deal with missing values or permutations of attributes).

SoftMealy is a system based on non-deterministic finite state automata (NDFA), and it was mainly conceived to induce wrappers for semistructured pages. In contrast to WIEN, SoftMealy can handle missing values. The system produces extraction rules by means of a bottom-up inductive learning algorithm that uses labelled training pages represented as an automaton defined on all the permutations of the input data. The states and the state transitions of the automaton correspond, respectively, to extracted data and resulting rules. SoftMealy is more expressive than WIEN, although it has an efficiency limit since it considers each possible permutation of the input data.

STALKER is a system for learning supervised wrappers. The training set is provided by a user that must specify the relevant data to be extracted: each page is associated with a sequence of tokens that are interpreted as landmarks for the extraction phase. The system yields an Embedded Catalog Tree (ECT), representing the structure of a page as a tree in which the root corresponds to the complete sequence of tokens and each internal node is associated with portions of the complete sequence. Each node specifies how to identify the associated subsequences, using two Simple Landmark Grammars (SLG) capable of recognizing the delimiters of such sequences. STALKER is well-suited to extracting data from hierarchically structured data sources, and it does not rely on the order of items (i.e. it can easily handle missing values). Moreover, since the ECT is an unordered tree, it deals with permutations of slots straightforwardly. In spite of its high flexibility, STALKER seems to suffer the generation of complex extraction rules when, in several cases, a simple extraction rule, based on the order of the slots, would be sufficient to extract the desired information.

RAPIER (Robust Automated Production of Information Extraction Rules) is a wrapper induction system for semistructured pages. It takes as input a document and a template indicating the data to be extracted from such a document, and outputs pattern matching rules according to a given template. To generate the patterns, RAPIER uses both syntactic and semantic features appearing, respectively, in a *tagger* (which assigns to each word a proper label) and a *dictionary* (designed to contain information about the semantic classes in the document under analysis). The extraction rules are composed of three parts: a *pre-compiler*, a *compiler* and a *post-compiler*. Each part is a pattern used to identify the target data and the delimiters for such data. The process of pattern creation starts with a specific pattern for each example and generalizes these patterns by considering each pair of the created patterns.

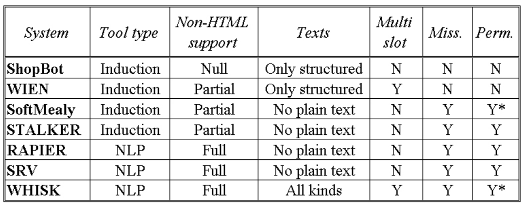
SRV (Sequence Rules with Validation) is a top-down relational learning algorithm. It works on a given set of labelled pages, and uses some features (related to the tokens appearing in the pages) to generate first-order logic extraction rules. The features considered by SRV are either simple or relational. Each simple feature maps a token to a categorical value (such as, length, font type, orthography), while relational features represent syntactic/semantic links among tokens. SRV generates the extraction rules using all the possible instances (i.e. sentences) in the document. Each instance, presented as an input to the classifier, is associated with a score indicating its suitability as a filler for the target slot. Extracted rules are highly expressive, since they can incorporate parts of speech, semantic classes etc.

The *WHISK* system can deal with all kinds of texts, since it exploits a syntactic analyzer and a semantic classifier. Given a training set of pages, WHISK generates regular expressions which are used to recognize the context of the relevant instances (i.e. sentences) and the delimiters of such instances. The system uses a covering algorithm to induce such regular expressions in a top-down fashion: the learning phase starts with the most general rule and continues by progressively specializing the available rules; the process stops when all the instances are covered by the set of rules. A major limitation of this system is the need for considering all the possible permutations (as well as SoftMealy).

4. Comparison

The table shown in Fig. 1 summarizes the main features of the systems so far described. The columns report, respectively:

– the technique exploited,
– the support for non-HTML pages,
– the documents that can be managed,
– the support for multi-slot extraction[1],
– the capability to deal with missing/distinct values,
– the capability to handle permutations of values.



System	*Tool type*	*Non-HTML support*	*Texts*	*Multi slot*	*Miss.*	*Perm.*
ShopBot	Induction	Null	Only structured	N	N	N
WIEN	Induction	Partial	Only structured	Y	N	N
SoftMealy	Induction	Partial	No plain text	N	Y	Y*
STALKER	Induction	Partial	No plain text	N	Y	Y
RAPIER	NLP	Full	No plain text	N	Y	Y
SRV	NLP	Full	No plain text	N	Y	Y
WHISK	NLP	Full	All kinds	Y	Y	Y*

Fig. 1. Different systems compared (Y* means that the training set must include all the possible permutations).

As we can see, the well complete system is WHISK, which performs on both semi-structured and unstructured text, and can deal with missing values and permutation of fields. However, it requires a "complete" training set, i.e. a set of examples including all the possible occurrences of values. STALKER and SRV do not require such complete training sets, but cannot deal with plain text; moreover, they do not perform multi-slot extraction. WIEN is able to perform multi-slot extraction, but requires highly structured pages. STALKER is the only system capable of dealing with pages exhibiting a highly complex structure; in particular, it is able to deal with pages containing information hierarchically structured into an indefinite number of levels.

References

[1] B. Adelberg. NoDoSE: A Tool for Semi-Automatically Extracting Semistructured Data from Text Documents. In *Proc. SIGMOD'98 Conf.*, 1998.

[2] A. Arocena and A.O. Mendelzon. Restructuring Documents, Databases, and Webs. In *Proc. ICDE'98 Conf.*, pages 24–33, 1998.

[3] R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. In *Proc. VLDB'01 Conf.*, pages 119–128, 2001.

[4] M. E. Califf and R. J. Mooney. Relational Learning of Pattern-Match Rules for Information Extraction. In *Proc. AAAI'99*, pages 328–334, 1999.

[5] V. Crescenzi and G. Mecca. Grammars Have Exception. *Information Systems*, 23(8):539–565, 1998.

[6] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proc. VLDB'01 Conf.*, pages 109–118, 2001.

[7] Line Eikvil. Information Extraction from World Wide Web - A Survey. Technical Report 945, Norweigan Computing Center, 1999.

[8] D. W. Embley et al. Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. *Data and Knowledge Engineering*, 31(3):227–251, 1999.

[9] D. Freitag. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, 39(2–3):233–272, 2000.

[10] J. Hammer, J. McHugh, and H. Garcia-Molina. Semistructured Data: The TSIMMIS Experience. In *Proc. 1st East-European Symposium on Advances in Databases and Information systems*, pages 1–8, 1997.

[11] C. N. Hsu and M. T. Dung. Wrapping Semistructured Web Pages with Finite-State Transducers. In *Proc. Conf. on Automatic Learning and Discovery*, 1998.

[12] N. Kusmerick. Wrapper Induction: efficiency and expressiveness. *Artificial Intellegence Journal*, 118(1–2):15–68, 2000.

[13] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A Brief Survey of Web Data Extraction Tools. *SIGMOD Records*, 31(2), 2002.

[14] I. Muslea, S. Minton, and C. Knoblock. Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.

[15] A. Sahuguet and F. Azavant. Building Intelligent Web Applications Using Lightweight Wrappers. *Data & Knowledge Engineering*, 36:283–316, 2001.

[16] S. Soderland. Learning Information Extraction Rules for Semistructured and Free Text. *Machine Learning*, 34(1–3):233–272, 1999.

[1] A wrapper performs *multi-slot extraction* if each extraction rule may return a sequence of fields (slots).