WEB DATA EXTRACTION SYSTEM

Robert Baumgartner
Institute for Information Systems
Vienna University of Technology and
Lixto Software GmbH, Austria
http://www.dbai.tuwien.ac.at/staff/baumgart/

Wolfgang Gatterbauer
Computer Science and Engineering
University of Washington, USA
http://www.cs.washington.edu/homes/gatter/

Georg Gottlob
Computing Laboratory
Oxford University, UK
http://web.comlab.ox.ac.uk/oucl/people/georg.gottlob.html

SYNONYMS

web data extraction toolkit, web information extraction system, wrapper generator, wrapper generator toolkit, web macros, web scraper.

DEFINITION

A web data extraction system is a software system that automatically and repeatedly extracts data from web pages with changing content and delivers the extracted data to a database or some other application. The task of web data extraction performed by such a system is usually divided into five different functions: (1) *web interaction*, which comprises mainly the navigation to usually pre-determined target web pages containing the desired information; (2) support for *wrapper generation and execution*, where a wrapper is a program that identifies the desired data on target pages, extracts the data and transforms it into a structured format; (3) *scheduling*, which allows repeated application of previously generated wrappers to their respective target pages; (4) *data transformation*, which includes filtering, transforming, refining, and integrating data extracted from one or more sources and structuring the result according to a desired output format (usually XML or relational tables); and (5) *delivering* the resulting structured data to external applications such as database management systems, data warehouses, business software systems, content management systems, decision support systems, RSS publishers, email servers, or SMS servers. Alternatively, the output can be used to generate new web services out of existing and continually changing web sources.

HISTORICAL BACKGROUND

The precursors of web data extraction systems were screen scrapers which are systems for extracting screen formated data from mainframe applications for terminals such as VT100 or IBM 3270. Another related issue are ETL methods (ETL = Extract, Transform, Load), which extract information from various business processes and feed it into a database or a data warehouse. A huge gap was recognized to exist between web information and the qualified, structured data as usually required in corporate information systems or as envisioned by the Semantic Web. In many application areas there has been a need to automatically extract relevant data from HTML sources and translate this data into a structured format, e.g., XML, or into a suitable relational database format.

A first and obvious solution was the evolution from screen scrapers to web scrapers, which can navigate to web pages and extract textual content. However, web scrapers usually lack the logic necessary to define highly structured output data as opposed to merely text chunks or textual snippets. Moreover, they are usually unable to collect and integrate data from different related sources, to define extraction patterns that remain stable in case of minor layout changes, to transform the extracted data into desired output formats, and to deliver the refined data into different kinds of applications. For this reason, specific research on web data extraction was needed and several academic projects and some commercial research projects on web data extraction were initiated before or around the year 2000. While the academic projects such as XWRAP [11], Lixto [2], Wargo [13], and the commercial systems RoboMaker of Kapow technologies[1] and WebQL of QL2 Software[2] focused on methods of strongly supervised "semi-automatic" wrapper generation, providing a wrapper designer with visual and interactive support for declaring extraction and formatting patterns, other projects were based on machine learning techniques. For example, WIEN [8], Stalker [12], and DEByE [9] focused on automatic wrapper induction from annotated examples. Some other projects focused on specific issues such as automatic generation of structured data sets from web pages [3] and particular aspects of navigating and interacting with web pages [1]. Finally, some systems require the wrapper designer to program the wrapper in a high level language (e.g. SQL-like languages for selecting data from a web page) and merely give visual support to the programmer; an example is the W4F system [14]. A number of early academic prototypes gave rise to fully fledged systems that are now commercially available. For example, Stalker gave rise to the Fetch Agent Platform of Fetch Technologies[3], the Lixto system gave rise to the Lixto Suite of the Lixto Software company[4], and the Wargo system evolved into the Denodo Platform [13]. A good and almost complete survey of web information extraction systems up to 2002 is given in [15].

[1] Kapow Technologies. http://www.kapowtech.com/
[2] QL2 Software Inc. http://www.ql2.com/
[3] Fetch Technologies, Inc. http://www.fetch.com/
[4] Lixto Software GmbH. http://www.lixto.com/

SCIENTIFIC FUNDAMENTALS

FORMAL FOUNDATIONS AND SEMANTICS OF DATA EXTRACTION. There are four main formal approaches to define (the semantics of) web wrappers: ► In the *functional approach*, a web wrapper is seen as a mapping *f* from the parse or DOM tree *T* of a web page to a set *f(T)* of sub-trees of *T*, where each sub-tree corresponds to an extracted data object. Another step specifies the relabling of extracted data to fit a previously defined output schema. ► The *logical approach* (see web data extraction: logical fundamentals) consists of the specification of a finite number of monadic predicates, i.e., predicates, that define sets of parse-tree nodes and that evaluate to true or false on each node of a tree-structured document. The predicates can be defined either by logical formulas, or by logic programs such as monadic datalog, which was shown to be equivalent in expressive power to monadic second-order logic (MSO) [6]. Note that languages such as XPATH or regular path expressions are subsumed by MSO. ► The *automata theoretic approach* relies on tree automata, which are a generalization of finite state automata from words to trees. The equivalence of the two logical approaches, MSO and monadic datalog, and the unranked query automata approach shows that the class of wrappers definable in these formalisms is quite natural and robust. Fortunately, this class is also quite expressive, as it accommodates most of the relevant data extraction tasks and properly contains the wrappers definable in some specifically designed wrapper programming languages [7]. ► Finally, the *textual approach* interprets web pages as text strings and uses string pattern matching for data extraction [12].

METHODS OF WRAPPER GENERATION. With regard to user involvement, three principal approaches for wrapper generation can be distinguished: (1) *Manual wrapper programming*, in which the system merely supports a user in writing a specific wrapper but cannot make any generalizations from the examples provided by the user; (2) *wrapper induction*, where the user provides examples and counterexamples of instances of extraction patterns, and the system induces a suitable wrapper using machine learning techniques; and (3) *semi-automatic interactive wrapper generation*, where the wrapper designer not only provides example data for the system, but rather accompanies the wrapper generation in a systematic computer-supported interactive process involving generalization, correction, testing, and visual programming techniques.

ARCHITECTURE. Figure 1 depicts a high-level view of a typical fully-fledged semi-automatic interactive web data extraction system. This system comprises several tightly connected components and interfaces three external entities: (1) *the Web*, which contains pages with information of interest; (2) a *target application*, to which the extracted and refined data will be ultimately delivered; and (3) *the user*, who interactively designs the wrapper.

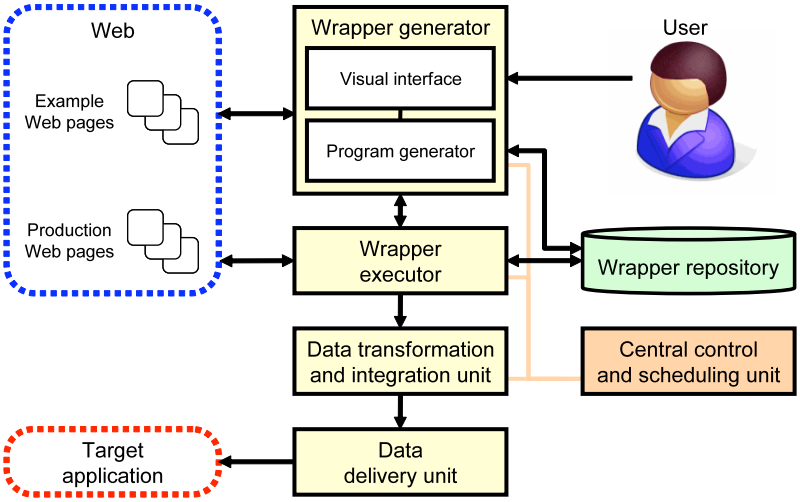

Fig. 1: Architecture of a typical web data extraction system

The wrapper generator supports the user during the wrapper design phase. It commonly has a *visual interface* that allows the user to define which data should be extracted from web pages and how this data should be mapped into a structured format such as XML. The visual interface itself can include several windows, such as: (i) a browser window that renders example web pages for the user; (ii) a parse tree window that shows the HTML parse tree of the current web page rendered by the browser; (iii) a control window that allows the user to view and control the overall progress of the wrapper design process and to input textual data (e.g. attribute names or XML tags); and (iv) a program window that displays the wrapper program constructed so far and allows the user to further adjust or correct the program.

The subunit that actually generates the wrapper is contained in what is here referred to as the *program generator*. This submodule interprets the user actions on the example web pages and successively generates the wrapper. Semi-automatic wrapper generators allow the user to either specify the URL of example web pages, which are structurally similar to the target web pages, or to navigate to such example pages. In the latter case, the navigation is recorded and can be automatically reproduced. The example pages are rendered in a browser window. Most wrapper generators also display the parse tree of the current page. In most systems, the wrapper designer can click on a node of the HTML parse tree, and the system immediately highlights the corresponding part of the example page in the browser, and vice versa. The designer can, thus, iteratively narrow down an example of a relevant data item after a few trials. Many web data extraction systems exhibit an XPATH-like path expression that precisely identifies the selected data item. The wrapper designer can then generalize this path expression by replacing some of the elements with wildcards. The result is a generalized pattern that matches several similar data items on similarly structured pages. Some systems (e.g. Denodo and Lixto) allow a wrapper designer to select data items directly on the rendered web page. The wrapper generator often also provides support for associating a name or

tag to each extraction pattern and for organizing patterns hierarchically. Systems based on wrapper induction allow the user to supply a large number of sample web pages with positive and/or negative examples of desired data items, from which the program generator tries to learn a generalized pattern. Systems for manual wrapper programming simply provide a visual environment for developing a wrapper in the underlying wrapper programming language, for testing the wrapper against example pages and for debugging. In addition, the visual interface usually allows an application designer to specify when a wrapper should be executed.

The *wrapper executor* is the engine that facilitates the deployment of the previously generated wrappers. Typically, a wrapper program comprises functions such as deep web navigation (e.g. the means to fill out forms), identification of relevant elements with declarative rules or procedural scripts, and generation of a structured output format (e.g. XML or relational tables). Typically, the wrapper executor can also receive additional input parameters such as a start URL, a predefined output structure (e.g. a particular XML schema), and special values for forms. In some approaches, the output includes metadata (e.g. verification alerts in case of violated integrity constraints) in addition to the actual data of interest.

The *wrapper repository* stores the generated wrappers together with metadata. In some approaches, wrapper templates and libraries of domain concepts can be reused and extended. Systems such as Dapper[5] offer a community-based extensible wrapper repository. In principle, systems can offer automatic wrapper maintenance and adaptation as part of the repository.

The *data transformation and integration unit* provides a means to streamline the results of multiple wrappers into one harmonized homogeneous result. This step includes data integration, transformation and cleaning, basically functions commnly found in the *Transform* step of ETL processes of data warehouses and mediation systems. Technologies and tools comprise query languages, visual data mapping, automatic schema matching and de-duplication techniques. Additionally, the data can be composed into desired result formats such as HTML, Excel or PDF.

The *data delivery unit* (capturing the *Load* step in the ETL philosophy) decides about the appropriate output channel such as email, Open Database Connectivity (ODBC) connection, or FTP. Advanced wrapper generation systems offer a number of system connectors to state-of-the-art databases and enterprise software such as Customer Relationship Management (CRM) or Enterprise Resource Planning (ERP), or partner with software providers in the area of Enterprise Application Integration (EAI) or Business Intelligence (BI) who offer such solutions.

The *central control and scheduling unit* is the heart of the processing engine. In general, synchronous and asynchronous extraction processes can be distinguished. A typical example of an asynchronous triggering is a real-time user request in a flight meta-search application. The request is queued, the respective wrappers are triggered, parameter mappings are performed, and the result is integrated and

[5] Dapper. http://www.dapper.net/

presented to the user in real-time. Sophisticated approaches offer a workflow-based execution paradigm, including parallelism, returning partial results, and interception points in complex booking transactions. On the other hand, market monitoring or web process integration are typical synchronous scenarios. An intelligent scheduler is responsible for distributing requests to the wrapper executor, iterating over a number of possible form inputs, and dealing with exception handling.

COMMERCIAL WRAPPER GENERATION SYSTEMS include:

Dapper. The Dapp Factory of Dapper offers fully server-based wrapper generation, and supports a community driven reusable wrapper repository. Dapper strongly relies on machine learning techniques for wrapper generation and concentrates exclusively on web pages that can be reached without deep web navigation. Users designing a wrapper label positive and negative examples and debug the result on a number of similarly structured web pages. Wrappers are hosted on the server and wrapper results can be queried via REST. Commercial APIs are offered for using Dapper in Enterprise scenarios.

Denodo[6]. The Denodo ITPilot, formerly known as Wargo, is a platform for creating and executing navigation and extraction scripts that are loosely tied together. It offers graphical wizards for configuring wrappers and allows DOM events to be processed while navigating web pages. Deep Web navigation can be executed in Internet Explorer, and the result pages are passed on to the extraction program. Furthermore, ITPilot offers some wrapper maintenance functionalities. Denodo additionally offers a tool called Aracne for document crawling and indexing.

Lixto. The Lixto Suite comprises the Lixto Visual Developer (VD), a fully visual and interactive wrapper generation framework, and the Java-based Lixto Transformation Server providing a scalable runtime and data transformation environment, including various Enterprise Application connectors. VD is ideally suited for dynamic Web 2.0 applications as it supports a visual Deep Web macro recording tool that makes it possible to simulate user clicks on DOM elements. Tightly connected with the navigation steps and hidden from the wrapper designer, the expressive language Elog is used for data extraction and linking to further navigation steps. VD is based on Eclipse and embeds the Mozilla browser.

Kapowtech. The Kapow RoboSuite (recently rebranded to Kapow Mashup Server) is a Java-based visual development environment for developing web wrappers. It embeds a proprietary Java browser, implements a GUI on top of a procedural scripting language, and maps data to relational tables. The RoboServer offers APIs in different programming languages for synchronous and asynchronous communication. In addition, variants of the Mashup Server for specific settings such as content migration are offered.

[6] Denodo. http://www.denodo.com

WebQL. QL2 uses a SQL-like web query language called WebQL for writing wrappers. By default, WebQL uses its own HTML DOM tree model instead of relying on a standard browser, but a loose integration with Internet Explorer is offered. The QL2 Integration Server supports concepts such as server clustering and HTML diffing. Furthermore, an IP address anonymization environment is offered.

KEY APPLICATIONS

Web market monitoring. Nowadays, a lot of basic information about competitors can be retrieved legally from public information sources on the Web, such as annual reports, press releases or public data bases. On the one hand, powerful and efficient tools for Extracting, Transforming and Loading (ETL) data from internal sources and applications into a Business Intelligence (BI) data warehouse are already available and largely employed. On the other hand, there is a growing economic need to efficiently integrate external data, such as market and competitor information, into these systems as well. With the World Wide Web as the largest single database on earth, advanced data extraction and information integration techniques as described in this paper are required to process this web data automatically. At the same time, the extracted data has to be cleaned and transformed into semantically useful formats and delivered in a "web-ETL" process into a BI system. Key factors in this application area include scalable environments to extract and schedule processing of very large data sets efficiently, capabilities to pick representative data samples, cleaning extracted data to make it comparable, and connectivity to data warehouses.

Web process integration. In markets such as the automotive industry, business processes are largely carried out by means of web portals. Business critical data from various divisions such as quality management, marketing and sales, engineering, procurement and supply chain management have to be manually gathered from web portals. Through automation, suppliers can dramatically reduce the cost while at the same time improving speed and reliability of these processes. Additionally, the automation means to leverage web applications to web services, and hence wrapper generation systems can be considered as an enabling technology for Service Oriented Architectures (SOA) as envisioned and realized in Enterprise Application Integration (EAI) and B2B processes. Key factors in this application area are workflow capabilities for the whole process of data extraction, transformation and delivery, capabilities to treat all kinds of special cases occurring in web interactions, and excellent support of the latest web standards used during secure transactions.

Mashups. Increasingly, leading software vendors have started to provide mashup platforms such as Yahoo! Pipes or IBM QEDWiki. A mashup is a web application that combines a number of different websites into an integrated view. Usually, the content is taken via APIs by embedding RSS or atom feeds similar to REST (Representational State Transfer). In this context, wrapper technology transforms legacy web applications to light-weight APIs that can be integrated in mashups in the same way. As a result, web mashup solutions no longer need to rely on APIs

offered by the providers of websites, but can rather extend the scope to the whole Web. Key factors for this application scenario include efficient real-time extraction capabilities for a large number of concurrent queries and detailed understanding of how to map queries to particular web forms.

FUTURE DIRECTIONS*

► *Generic web wrapping*. Without explicit semantic annotations on the current Web (compare Semantic Web), data extraction systems that allow general web wrapping will have to move towards fully automatic wrapping of the existing World Wide Web. Important research challenges are (i) how to optimally bring semantic knowledge into the extraction process, (ii) how to make the extraction process robust (in particular, how to deal with false positives), and (iii) how to deal with the immense scaling issues. Today's web harvesting and automated information extraction systems show much progress (see e.g. [4][10]), but still lack the combined recall and precision necessary to allow for very robust queries. ► A related topic is that of *auto-adapting wrappers.* Most wrappers today depend on the tree structure of a given web page and suffer from failure when the layout and code of web pages change. Auto-adapting wrappers, which are robust against such changes, could use existing knowledge of the relations on previous versions of web page in order to automatically "heal" and adapt the extraction rules to the new format. The question here is how to formally capture change actions and execute the appropriate repair actions. ► *Wrapping from visual layouts.* Whereas web wrappers today dominantly focus on either the flat HTML code or the DOM tree representation of web pages, recent approaches aim at extracting data from the CSS box model and, hence, the visual representation of web pages [5]. This method can be particularly useful for layout-oriented data structures such as web tables and allows to create automatic and domain-independent wrappers which are robust against changes of the HTML code implementation. ► *Data extraction from non-HTML file formats.* There is a substantial interest from industry in wrapping documents in formats such as PDF and PostScript. Wrapping of such documents must be mainly guided by a visual reasoning process over white space and Gestalt theory, which is substantially different from web wrapping and will, hence, require new techniques and wrapping algorithms including concepts borrowed from the document understanding community. ► *Learning to deal with web interactions.* As web pages are becoming increasingly dynamic and interactive, efficient wrapping languages have to make it possible to record, execute and generalize macros of web interactions and, hence, model the whole process of workflow integration. An example of such a web interactions is a complicated booking transaction. ► *Web form understanding and mapping.* In order to automatically or interactively query deep web forms, wrappers have to learn the process of filling out complex web search forms and the usage of query interfaces. Such systems have to learn abstract representation for each search form and map them to a unified meta form and vice versa, taking into account different form element types, contents and labels.

CROSS REFERENCES

web harvesting, web information extraction, data integration, logical fundamentals of web data extraction, snippet, service oriented architecture, datalog, data mining, wrapper, wrapper generator, wrapper induction, wrapper maintenance, wrapper stability, web services, business intelligence, enterprise application integration, XML, Xpath/Xquery, Semantic Web, ETL, screen scraper, data warehouse,

RECOMMENDED READING*

[1] V. Anupam, J. Freire, B. Kumar and D. Lieuwen. Automating web navigation with the WebVCR. *Computer Networks*, 33(1-6): 503-517, 2000.

[2] R. Baumgartner, S. Flesca and G. Gottlob. Visual web information extraction with Lixto. *VLDB*, 2001.

[3] V. Crescenzi, G. Mecca and P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. *VLDB*, 2001.

[4] O. Etzioni, M.J. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D.S. Weld, and Y. Yates. Web-scale information extraction in KnowItAll: (preliminary results), *WWW*, 2004.

[5] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl and B. Pollak. Towards domain-independent information extraction from web tables. *WWW*, 2007.

[6] G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *JACM*, 51(1): 74-113, 2002.

[7] G. Gottlob and C. Koch. A formal comparison of visual web wrapper generators. *SOFSEM*, 2006.

[8] N. Kushmerick, D. S. Weld and R.B. Doorenbos. Wrapper induction for information extraction. *IJCAI*, 1997.

[9] A.H.F. Laender, B.A. Ribeiro-Neto and A.S. da Silva. DEByE - Data extraction by example. *Data and Knowledge Engineering*, 40(2): 121-154, 2000.

[10] B. Liu, R.L. Grossman and Y. Zhai. Mining web pages for data records. *IEEE Intelligent Systems*, 19(6):49-55, 2004.

[11] L. Liu, C. Pu and W. Han. XWRAP: An XML-enabled wrapper construction system for web information sources. *ICDE*, 2000.

[12] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. *Autonomous Agents and Multi-Agent Systems*, 4(1/2): 93-114, 2001.

[13] A. Pan, J. Raposo, M. Álvarez, P. Montoto, V. Orjales, J. Hidalgo, L. Ardao, A. Molano and Á. Viña. The Denodo data integration platform. *VLDB*, 2002.

[14] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data and Knowledge Engineering*, 36:(3) 283-316, 2001.

[15] S. Kuhlins and R. Tredwell. Toolkits for Generating Wrappers: A Survey of Software Toolkits for Automated Data Extraction from Websites. NODe 2002, LNCS:2591, 2003.