Web Data Extraction, Applications and Techniques: A Survey

EMILIO FERRARA and GIACOMO FIUMARA
University of Messina
and
ROBERT BAUMGARTNER
Lixto Software GmbH, Austria

The World Wide Web contains a huge amount of unstructured and semi-structured information, that is exponentially increasing with the coming of the Web 2.0, thanks to User-Generated Contents (UGC). In this paper we intend to briefly survey the fields of application, in particular enterprise and social applications, and techniques used to approach and solve the problem of the extraction of information from Web sources: during last years many approaches were developed, some inherited from past studies on Information Extraction (IE) systems, many others studied ad hoc to solve specific problems.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software—*Expert system tools and techniques*

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: Semantic Web, Information Extraction, Data Mining

1. INTRODUCTION

This work is a brief survey on the problem of the Web data extraction, in particular concerning *fields of application*, *approaches* and *techniques* developed during the years. We tried to change the approach and the point of view used in the past to survey this discipline: many works presented a list of tools, reporting a feature-based analysis or benchmarking them and reporting results. Another standard approach used, was to track a list of approaches and classify each tool analyzed. Many of these works are very solid starting points in the study of this area. Rather, our contribution tackles the problem of surveying from a different point of view.

In particular we focus this survey on fields of application of tools developed with classic and novel techniques covered by the discipline of the Web data extraction; at the best of our knowledge, this is the first time a Web data extraction survey deeply

E-mail addresses: emilio.ferrara@unime.it (E. Ferrara); giacomo.fiumara@unime.it (G. Fiumara); robert.baumgartner@lixto.com (R. Baumgartner).

analyzes application fields. We want to cover in particular enterprise, social and scientific applications that are strictly interconnected with Web data extraction tasks and discover which fields have already been approached (e.g. advertising, software engineering, enterprise solutions, business and competitive intelligence, etc.) and which are going to be, looking ahead the immediate future (e.g. bio-informatics, web harvesting, etc.). This work is structured as follows: in Section 2 we will take a look at related work, in Section 3 we track a complete profile of Web data extraction systems, in Section 4 we classify fields of application of Web data extraction, focusing, in particular, on enterprise and social applications, with an eye to the future. In Section 5 we survey approaches and techniques relating wrapper generation, induction and maintenance, plus other notable approaches like spatial reasoning, etc. Finally, Section 6 concludes including some recommendations, contributions and suggestions for future work.

2. RELATED WORKS

The Computer Science scientific literature counts many valid surveys on the Web data extraction problem: Laender et al. [Laender et al. 2002], in 2002, presented a notable survey, offering a rigorous *taxonomy* to classify Web data extraction systems. They introduced a set of criteria and a qualitative analysis of various Web data extraction tools.

In the same year Kushmerick [Kushmerick 2002] tracked a profile of *finite-state approaches* to the problem, including analyzing wrapper induction and maintenance, natural language processing and hidden Markov models. Kuhlins and Tredwell [Kuhlins and Tredwell 2003] surveyed tools for generating wrappers already in 2003: information could not be up-to-date but analyzing the approach is still very interesting. Again on the wrapper induction problem, Flesca et al. [Flesca et al. 2004] and Kaiser and Miksch [Kaiser and Miksch 2005] surveyed approaches, techniques and tools. The latter in particular modeled a representation of an Information Extraction system architecture.

Chang et al. [Chang et al. 2006] introduced a *tri-dimensional categorization* of Web data extraction systems, based on task difficulties, techniques used and degree of automation. Fiumara [Fiumara 2007] applied these criteria to classify four new tools that are also presented here. Sarawagi published an illuminating work on Information Extraction [Sarawagi 2008]: anybody who intends to approach this discipline should read it. To the best of our knowledge, the newest work from Baumgartner et al. [Baumgartner et al. 2009] is a short survey on the state-of-the-art of the discipline.

3. WEB DATA EXTRACTION SYSTEMS

3.1 Definition

We can generically define a Web data extraction system as *a sequence of procedures that extracts information from Web sources* [Laender et al. 2002]. From this generic definition, we can infer two fundamental aspects of the problem:

—Interaction with Web pages

—Generation of a wrapper

Baumgartner et al. [Baumgartner et al. 2009] define a Web data extraction system, as 'a software extracting, *automatically* and *repeatedly*, data from Web pages with changing contents, and that *delivers extracted data* to a database or some other application'.

This is the definition that better fits the modern view of the problem of the Web data extraction as it introduces three important aspects:

—Automation and scheduling

—Data transformation, and the

—Use of the extracted data

The following five points cover techniques used to solve the problem of Web data extraction.

3.1.1 *Interaction with Web pages.* The first phase of a generic Web data extraction system is the *Web interaction* [Wang et al. 2000]: Web sources, usually represented as Web pages, but also as RSS/Atom feeds [Hammersley 2005], Microformats [Khare and Çelik 2006] and so on, could be surfed by users, both in visual and textual mode, or just simply inputted to the system by the URL of the document(s) containing the information.

Some commercial systems, Lixto for first but also Kapow Mashup Server (described below), include a *Graphical User Interface* for fully *visual and interactive navigation* of HTML pages, integrated with data extraction tools.

The state-of-the-art is represented by systems that support the extraction of data from pages reached by *deep Web navigation* [Baumgartner et al. 2005], i.e. simulating the activity of users clicking on DOM elements of pages, through macros or, more simply, filling HTML forms.

These systems also support the extraction of information from *dynamically generated Web pages*, usually built at run-time as a consequence of the user request, filling a template page with data from some database. The other kind of pages are commonly called *static Web pages*, because of their static content.

3.1.2 *Generation of a wrapper.* Just for now, we generically define the concept of wrapper as a procedure extracting unstructured information from a source and transforming them into structured data [Zhao 2007; Irmak and Suel 2006]. A Web data extraction system must implement the support for wrapper generation and wrapper execution. We will cover approaches and techniques, used by several systems, later.

3.1.3 *Automation and scheduling.* The *automation* of page access, localization and extraction is one of the most important features included in last Web data extraction systems [Phan et al. 2005]: the *capability to create macros* to execute multiple instances of the same task, including the possibility to *simulate the click stream of the user*, *filling forms* and *selecting menus and buttons*, the support for AJAX technology [Garrett 2005] to handle the asynchronous updating of the page, etc. are only some of the most important automation features.

Also the scheduling is important, e.g. if an user wants to extract data from a news website updated every 5 minutes, many of the last tools let him to setup a scheduler, working like a *cron*, launching macros and executing scripts automatically and periodically.

3.1.4 *Data transformation.*

Information could be wrapped from multiple sources, which means using different wrappers and also, probably, obtaining different structures of extracted data. The steps between extraction and delivering are called *data transformation*: during these phases, such as *data cleaning* [Rahm and Do 2000] and *conflict resolution* [Monge 2000], users reach the target to obtain homogeneous information under a unique resulting structure.

Most powerful Web data extraction systems provide tools to perform *automatic schema matching* from multiple wrappers [Rahm and Bernstein 2001], then packaging data into a desired format (e.g. a database, XML, etc.) to make it possible to *query data*, *normalize structure* and *de-duplicate tuples*.

3.1.5 *Use of extracted data.*

When the extraction task is complete, and acquired data are packaged in the needed format, these information are ready to be used; the last step is to deliver the package, now represented by structured data, to a managing system (e.g. a native XML DBMS, a RDBMS, a data warehouse, a CMS, etc.). In addition to all the specific fields of application covered later in this work, acquired data can be also generically used for analytical or statistical purposes [Berthold and Hand 1999] or simply to republish them under a structured format.

3.2 Classification criteria

3.2.1 *A taxonomy for characterizing Web data extraction tools.*

Laender et al. [Laender et al. 2002] presented a *widely accepted taxonomy* to classify systems, according to techniques used to generate wrappers:

3.2.1.1 *Languages for Wrapper Development:.*

Before the birth of some languages specifically studied for *wrapper generation* (e.g. Elog, the Lixto Web extraction language [Baumgartner et al. 2001a]) extraction systems relied on standard scripting languages, like Perl, or general purpose programming languages, like Java to create the environment for the wrapper execution.

3.2.1.2 *HTML-aware Tools:.*

Some tools rely on the *intrinsic formal structure of the HTML* to extract data, using HTML tags to build the DOM tree (e.g. RoadRunner [Crescenzi et al. 2001], Lixto [Baumgartner et al. 2001b] and W4F [Sahuguet and Azavant 1999]). Boronat [Boronat 2008], in his Master Thesis, analyzed and compared performances of common Web data extraction tools.

3.2.1.3 *NLP-based Tools:.*

Natural Language Processing techniques were born in the context of *Information Extraction* (IE) [Winograd 1972; Berger et al. 1996; Manning and Schütze 1999]. They were applied to the Web data extraction problem in order to solve specific problems such as the extraction of facts from speech transcriptions in forums, email messages, newspaper articles, resumes etc.

3.2.1.4 *Wrapper Induction Tools:.*

These tools generate rule-based wrappers, automatically or semi-automatically: usually they rely on *delimiter-based* extraction criteria inferred from *formatting features* [Anton 2004].

3.2.1.5 *Modeling-based Tools:.* Relying on *a set of primitives* to compare with the structure of the given page, these tools can find one or more objects in the page matching the primitive items [Embley et al. 1999; Gatterbauer et al. 2007]. A strong domain knowledge is needed, but it is a good approach for the extraction of data from Web sources based on *templates* and *dynamically generated* pages.

3.2.1.6 *Ontology-based Tools:.* These techniques do not rely on the page structure, but *directly on the data*. Actually, ontologies can be applied successfully on specific well-known domain applications, e.g. social networks and communities [Mika 2007], bio-informatics [Hu et al. 2004], etc. Some works try to apply the ontological approach to generic domains of Web data extraction [Han 2002] or to tables [Tanaka and Ishida 2006].

3.2.2 *Qualitative analysis criteria.* Laender et al. [Laender et al. 2002] remarked that this taxonomy is not intended to *strictly classify* a Web data extraction system, because it is common that some tools fit well in two or more groups. Maybe for this reason, they extended these criteria including:

3.2.2.1 *Degree of automation:.* Just determines the amount of *human effort* needed to run a Web data extraction tool.

3.2.2.2 *Support for complex objects:.* Nowadays, Web pages are based on the *rich-content paradigm*, so objects included in the Web sources could be complex. Only some systems can handle these kind of data.

3.2.2.3 *Page contents:.* Page contents can be distinguished in two categories: *semi-structured text* and *semi-structured data*. The first family fits better with NLP-based tools and Ontology-based tools, the latter with the others.

3.2.2.4 *Ease of use:.* Availability of a *Graphical User Interface* (GUI) is a must for last generation tools. Platforms often feature *wizards to create wrappers*, WYSIWYG editor interfaces, integration with Web browsers etc. Lixto [1], Denodo [2], Kapow Mashup Server [3], WebQL [4], Mozenda [5], Visual Web Ripper [6] all use advanced GUI to ease the user experience.

3.2.2.5 *XML output:.* XML is simply the standard, according to W3C [7], for the *semantic Web representation* of data. The capability to output the extracted data in XML format nowadays is not more an optional, at least for commercial software.

3.2.2.6 *Support for Non-HTML sources:.* NLP-based tools fit better in this domain: this is a great advantage, because a very large amount of data is stored in the Web in semi-structured texts (emails, documentations, logs, etc.)

[1] http://www.lixto.com/
[2] http://www.denodo.com/
[3] http://kapowtech.com/
[4] http://www.ql2.com/
[5] http://www.mozenda.com
[6] http://www.visualwebripper.com/
[7] http://www.w3.org/

3.2.2.7 *Resilience and adaptiveness:.* Web sources are usually *updated without any forewarning*, i.e. the frequency of updates is not known, so systems that generate wrappers with an high degree of *resilience* show better performances. Also the *adaptiveness* of the wrapper, moving from a specific Web source to another, within the same domain, is a great advantage.

4. APPLICATIONS

On the one hand the Web is moving to semantics and enabling machine-to-machine communication: it is a slow, long term evolution, but it has started, in fact. Extracting data from Web sources is one of the most important steps of this process, because is the key to build a solid level of reliable semantic information. On the other hand, Web 2.0 extends the way humans consume the Web with social networks, rich client technologies, and the consumer as producer philosophy. Hence, new evolvements put further requirements on Web data extraction rules, including to understand the logic of Web applications.

In the literature of the Web data extraction discipline, many works cover approaches and techniques adopted to solve some particular problems related to a single or, sometimes, a couple of fields of application. The aim of this section, at the best of our knowledge, for the first time, is to survey and analyze the greatest possible number of applications that are strictly interconnected with Web data extraction tasks. In the following, we describe a taxonomy in which key application fields, heavily involved with data extraction from Web sources, are divided into two families, *enterprise applications* and *social applications.*

4.1 Enterprise applications

We classify here software applications and procedures with a direct, subsequent or final *commercial scope.*

4.1.1 *Context-aware advertising.* Thanks to Applied Semantic, Inc.[8] first, and Google, who bought their 'AdSense' advertising solution later, this field captured a great attention. The main underlying principle is to present, to the final user, commercial *thematized advertisements* together with the content of the Web page the user is reading, ensuring a potential *increase of the interest* in the ad.

This aim can be reached *analyzing the semantic content* of the page, *extracting relevant information*, both in the structure and in the data, and then contextualizing the ads content and placement in the same page.

Contextual advertising, compared to the old concept of Web advertising, represents a *intelligent approach* to provide useful information to the user, statistically more interested in thematized ads, and a better source of income for advertisers.

4.1.2 *Customer care.* Usually medium-big-sized company, with customers support, receives a lot of unstructured information like emails, support forum discussions, documentation, shipment address information, credit card transfer reports, phone conversation transcripts, etc.: the capability of extracting these information *eases classifying them*, *inferring underlying relationships*, *populating own structured databases and ontologies*, etc.

[8]http://www.appliedsemantics.com/

Actually NLP-based techniques are the best approach to solve these problems.

4.1.3 *Database building.* In the Web marketing sector this is a key concept: generically we can define database building as the activity of *building a database* of information *about a particular domain*. Fields of application are countless: financial companies could be interested in *extracting financial data* from the Web, e.g. scheduling these activities to be executed *automatically and periodically*.

Also the real estate market is very florid: acquiring data from multiple Web sources is an important task for a real estate company, for comparison, pricing, co-offering, etc.

Companies selling products or services probably want to *compare their pricing with other competitors*: products pricing data extraction is an interesting application of Web data extraction systems. Finally we can list other related tasks, obviously involved in the Web data extraction: duplicating an on-line database, extracting dating sites information, capturing auction information and prices from on-line auction sites, acquiring job postings from job sites, comparing betting information and prices, etc.

4.1.4 *Software Engineering.* Extracting data from websites became interesting also for Software Engineering: Web 2.0 is usually strictly related to the concept of Rich Internet Applications (RIAs), Web applications characterized by an *high degree of interaction and usability*, inherited from the similarity to desktop applications. Amalfitano et al. [Amalfitano et al. 2008] are developing a Reverse Engineering approach to abstract Finite States Machines representing the *client-side behavior* offered by RIAs.

4.1.5 *Business Intelligence and Competitive Intelligence.* Baumgartner et al. [Baumgartner et al. 2005; Baumgartner et al. 2009; Baumgartner et al. 2010] deeply analyzed how to apply Web data extraction techniques and tools to improve the process of acquiring market information. A solid layer of knowledge is fundamental to optimize the decision-making activities and an huge amount of public information could be retrieved on the Web. They illustrate how to acquire these unstructured and semi-structured information; using the Lixto Suite to access, extract, clean and deliver data, it is possible to gather, transform and obtain information useful to business purposes. It is also possible to integrate these data with other common platforms for Business Intelligence, like SAP [9] or Microsoft Analysis Services [Melomed et al. 2006].

Wider, the process of gathering and analyzing information for business purposes is commonly called Competitive Intelligence, and is strictly related to data mining [Han and Kamber 2000]. Zanasi [Zanasi 1998] was the first to introduce the possibility of acquiring these data, through data mining processes, on public domain information. Chen et al. [Chen et al. 2002] developed a platform, that works more like a spider than like a Web data extraction system, which represents a useful tool to support operations of competitive intelligence providing data from the Web.

In BI scenarios the main requirements include scalability and efficient planning strategies to extract as much data as possible with the smallest number of possible

[9]http://www.sap.com

resources in time and space.

4.1.6 *Web Process Integration and Channel Management.* In the Web of today data is often available via APIs (e.g. refer to *Programmable Web*[10]) , light-weight and heavy-weight Web service techniques. Nevertheless, the larger amount of data is primarily available in semi-structured formats such as HTML. To use Web data in Enterprise Applications and service-oriented architectures, it is essential to provide means for automatically turning Web Applications and Web sites into Web Services, allowing structured and unified access to heterogeneous sources. This includes to understand the logic of the Web application, to fill out form values, and to grab relevant data.

In a number of business areas, Web applications are predominant among business partners for communication and business processes. Various types of processes are carried out on Web portals, covering activities such as purchase, sales, or quality management, by manually interacting with Web sites. Typical vertical examples, where Web data extraction proves useful include channel management in the travel industry (like automating the regular offerings of rooms on hotel portals with bi-directional web connectors), re-packaging complex Web transactions to Web services and consequently to other devices, as well as automating communication of automotive suppliers with automotive companies.

Tools for wrapper generation pave the way for *Web Process Integration* and help enable the *Web of Services*, i.e. the seamless integration of Web applications into a corporate infrastructure or service oriented landscape by generating Web services from given Web sites [Baumgartner et al. 2010]. Web process integration can be understood as front-end and "outside-in" integration: integrate cooperative and non-cooperative sources without the need for information provider to change their backend. Additional requirements in such scenarios include to support a large number of users and real-time parametrized Web queries and support of complex Web transactions.

4.1.7 *Functional Web Application Testing.* Testing and Quality Management are essential parts of the life-cycle of software. Facettes of testing are manifold, including functional tests, stress/load tests, integration tests, and testing against specifications, to name a few. Usually, the strategy is to automate a large percentage of functional tests and execute test runs as part of nightly builds as regression tests. Such tests occur at various levels, e.g. testing the system functionality as a blackbox via APIs, or testing the system on the GUI level simulating either the user's steps or creating a model of possible application states.

In today's world of Software-as-a-Service platforms and Web oriented architectures, Web application testing plays an important role. One aspect is simulating the user's path through the application logic. Web data extraction and navigation techniques are ideally suited to record test scripts in a simple VCR-like manner. Robust identification criteria can be created exploiting the tree structure or visual structure of the page. Typical actions in such test scripts include to set/get values of form fields, picking dates, checkpoints to compare values, and following different branches depending on the given page. Due to automation, every step can be

[10]http://www.programmableweb.com/

parametrized and a test script executed in variations.

The requirements for tools in the area of Web application testing are to deal well with Ajax/dynamic HTML, to create robust test scripts, to efficiently maintain test scripts, to execute test runs and create meaningful reports, and, unlike other application areas, the support of multiple state-of-the-art browsers in various versions is an absolute must. One widely used open source tool for Web application testing is *Selenium*[11].

4.2 Social applications

One can say that *social* is the engine of Web 2.0: many websites evolved into Web applications built around users, letting them to create a *Web of links between people*, to share thoughts, opinions, photos, travel tips, etc. Here we are mainly listing all these kind of applications born and grown in the Web and through the Web, thanks to User-Generated Contents (UGC), built from users for users.

4.2.1 *Social networks.* Social networks are the most important expression of *change in the use of the World Wide Web*, and are often considered a key step of the evolution to Web 2.0: millions of people creating a *digital social structure*, made of nodes (individuals, entities, groups or organizations) and connections (i.e. ties), - representing relationships between nodes, sometimes implementing hierarchies - sharing personal information, relying on platforms of *general purpose* (e.g. Facebook[12], MySpace[13], etc.) or *thematized* (e.g. Twitter[14] for micro-blogging, Flickr[15] for photo-sharing, etc.), all sharing a common principle: *on-line socialization.*

Social networks attracted an enormous attention, by both academic and industries, and many works studied almost every aspect of the phenomenon: extracting relevant data from social networks is a new interesting problem and field of application for Web data extraction systems.

Actually does not exist a tool specifically studied to approach and solve this problem: people are divided by ethics on mining personal data from social networks. Regardless of moral disputes, interesting applications of the Web data extraction from social networks could be: *acquiring information* from relationships between nodes, studying *distribution of ties* and verifying the theory of six degrees of separation (i.e. '*the Human Web*') [Milgram 1967], analyzing statistical data, finding heuristics, *inferring information*, etc.

4.2.2 *Social bookmarks.* Another form of social application is the social bookmarking, a new kind of *knowledge sharing*: users post links to Web sources of interest, into platforms with the capability of creating *folksonomies*, collaboratively tagging contents. Extracting relevant information from social bookmarks should be faster and easier than in other fields: HTML-aware and model-based extraction systems should fit very well with the semi-structured templates used by most common social bookmarking services. Sometimes data are distributed under struc-

[11]http://seleniumhq.org/
[12]http://www.facebook.com/
[13]http://www.myspace.com/
[14]http://twitter.com/
[15]http://www.flickr.com/

tured formats like RSS/Atom so acquiring these information is easier than with traditional HTML sources.

4.2.3 *Comparison shopping.* One of the most appreciated Web social services is the *comparison shopping*, through platforms with the capability to *compare products or services*, going from simple prices comparison to features comparison, technical sheets comparison, user experiences comparison, etc. These services heavily rely on Web data extraction, using websites as sources for data mining and a custom internal engine to make possible the comparison of similar items. Many Web stores today also offer *personalization forms* that make the extraction tasks more difficult: for this reason many last-generation commercial Web data extraction systems (e.g. Lixto, Kapow Mashup Server, UnitMiner[16], Bget[17]) provide support for deep navigation and dynamic content pages.

4.2.4 *Mashup Scenarios.* Today, leading software vendors provide mashup platforms (such as Yahoo! Pipes or Lotus Mashups) and establish mashup communication standards such as EMML[18]. A mashup is a Web site or Web application that combines a number of Web sites into an integrated view. Usually, the content is taken via APIs, embedding RSS or Atom Feeds in a REST-like way. With wrapper technology one can leverage legacy Web applications to light-weight APIs such as REST that can be integrated in mashups in the same fashion. Web Mashup Solutions no longer need to rely on APIs offered by the providers of sites, but can extend the scope to the whole Web. In particular, the deep Web gets accessible by encapsulating complex form queries and application logic steps into the methods of a Web Service. End users are put in charge to create their own views of the Web and embed data into other applications ("consumers as producers"), usually in a light-weight way. This results in "situational applications", possibly unreliable and unsecure applications, that however help to solve an urgent problem immediately. In Mashup Scenarios, one important requirement to Web data extraction tools is the ease of use for non-technical content managers, to give them the possibility to create new Web connectors without help of IT.

4.2.5 *Opinion sharing.* Related to comparison shopping, the opinion sharing represents its evolution: users want to *express opinions* on *products*, *experiences*, *services* they enjoyed, etc. The most common form of opinion sharing is represented by blogs, containing articles, reviews, comments, tags, polls, charts, etc. All these information usually lack of structure, so their extraction is a huge problem, also for current systems, because of the billions of Web sources now available. Sometimes model-based tools fit good, taking advantage of common templates (e.g. Wordpress[19], Blogger[20], etc.), other times natural language processing techniques fit better. Kushal et al. [Dave et al. 2003] approached the problem of opinion extraction and subsequent semantic classification of reviews of products.

Another form of opinion sharing in semi-structured platforms is represented by

[16]http://www.qualityunit.com/unitminer/
[17]http://www.bget.com/
[18]http://www.openmashup.org
[19]http://wordpress.org/
[20]https://www.blogger.com/

Web portals that let users to write unmoderated opinions on various topics.

4.2.6 *Citation databases.* Citation databases building is one of the most intensive Web data extraction fields of application: CiteSeer[21], Google Scholar[22], DBLP[23] and Publish or Perish[24] are brilliant examples of applying Web data extraction to approach and solve the problem of *collect digital publications*, extract relevant data, i.e. *references and citations*, and build a structured database, where users can perform searches, comparisons, count of citations, cross-references, etc.

4.2.7 *Web Accessibility.* Techniques for automatic data extraction and document understanding are extremely helpful in making Web pages more accessible to blind and partial-sighted users.

Today's solution approaches are inefficient to overcome the problem. The first approach, screen-reader usage, is optimized for native client interfaces, and not well equipped to deal with the presentation, content and interactions in Web 2.0 - such as understanding the reading order, telling the user what a date picker means, or jumping from one forum post to the next. The second approach, the Web Accessibility Initiative, is no doubt absolutely necessary and has defined valuable concepts, such as ARIA roles assignable to GUI elements. However, due to the additional investment at best such guidelines are applied in governmental sites.

Approaches such as ABBA [Fayzrakhmanov et al. 2010] overcome these limitations. In the ABBA approach, a Web page is transformed into a formal multi-axial semantic model; the different axes offer means to reason on and serialize the document by topological, layout, functional, content, genre and saliency properties. A blind person can navigate along and jump between these axes to skip to the relevant parts of a page. E.g., the presentational axis contains transformed visual cues, allowing the user to list information in the order of visual saliency.

4.3 A glance on the future

4.3.1 *Bio-informatics and Scientific Computing.* A growing field of application of the Web data extraction is *bio-informatics*: on the World Wide Web it is very common to find medical sources, in particular regarding *bio-chemistry* and *genetics*. Bio-Informatics is an excellent example of the application of scientific computing – refer e.g. to [Descher et al. 2009] for a selected scientific computing project.

Plake et al. [Plake et al. 2006] worked on PubMed[25] - the biggest repository of medical-scientific works that covers a broad range of topics - extracting information and relationships to create a graph; this structure could be a good starting point to proceed in extracting data about *proteins* and *genes*, e.g. connections and interactions among them: these information can be usually found, not in Web pages, rather they are available in PDF or Postscript format. In the future, Web data extraction should be extensively used also to these documents: approaches to solve this problem are going to be developed, inherited, both from Information Extrac-

[21]http://citeseer.ist.psu.edu/
[22]http://scholar.google.com/
[23]http://www.informatik.uni-trier.de/ley/db
[24]http://www.harzing.com/pop.html
[25]http://www.pubmed.com/

tion and Web data extraction systems, because of the semi-structured format of PostScript-based files. On the other hand, web services play a dominant role in this area as well, and another important challenge is the intelligent and efficient querying of Web services as investigated by the ongoing SeCo project [26].

4.3.2 *Web harvesting.* One of the most attractive future applications of the Web data extraction is *Web Harvesting* [Weikum 2009]: Gatterbauer [Gatterbauer 2009] defines it as 'the process of gathering and integrating data from various heterogeneous Web sources'. The most important aspect (although partially different from specific Web data extraction) is that, during the last phase of data transformation, the amount of gathered data is *many times greater* than the extracted data. The work of filtering and refining information from Web sources ensures that extracted data lie in the domain of interest and are relevant for users: this step is called *integration*. The Web harvesting remains an open problem with large margin of improvement: because of the billions of Web pages, it is a computational problem, also for restricted domains, to crawl enough sources from the Web to build a *solid ontological base*. There is also a human engagement problem, correlated to the degree of automation of the process: when and where humans have to interact with the system of Web harvesting? Should be a fully automatic process? What degree of precision can we accept for the harvesting? All these questions are still open for future works. Projects such as the DIADEM [27] at Oxford University tackle the challenge for fully automatic generation of wrappers for restricted domains such as real estate.

5. TECHNIQUES

5.1 Used approaches

The technique first used to extract data from Web pages were inherited from Information Extraction (IE) approaches.

Kaiser and Miksch [Kaiser and Miksch 2005] categorized them into two groups: *learning techniques* and *knowledge engineering techniques*.

Sarawagi [Sarawagi 2008] calls them *hand-coded* or *learning-based* approach and *rule-based* or *statistical* approach respectively; these definitions better explain the same concepts: the first method is used to develop a system that requires human expertise to define rules (usually *regular expressions* or *program snippets*) for performing the extraction. Both in hand-coded and learning-based approaches domain expertise is needed: people writing rules and training the system must have programming experience and a good knowledge of the domain and tasks of the system.

Also in some approaches of the latter family, in particular the *rule-based* ones, a strong familiarity with both the requirements and the functions is needed, so the human engagement is essential. Statistical methods are more effective and reliable in domains of unstructured data (like natural language processing problems, facts extraction from speeches, and automated text categorization [Sebastiani 2002]).

[26]http://www.search-computing.it/
[27]http://web.comlab.ox.ac.uk/projects/DIADEM/

5.2 Wrappers

A wrapper is a procedure that implements a family of algorithms, that *seek* and *find* the information the user needs to extract from an unstructured source, and *transform* them into structured data, *merging* and *unifying* these information for *future processing.*

A wrapper life-cycle starts with its *generation*: it could be described and implemented *manually*, e.g. using regular expressions, or in an inductive way; *wrapper induction* [Kushmerick 1997] is one of the most interesting aspects of this discipline, because introduces high level automation algorithms implementation; we can also count on hybrid approaches that make possible for users to generate semi-automatic wrappers thanks to visual interfaces. Web pages change without forewarning, so *wrapper maintenance* is an outstanding aspect for ensuring the regular working of wrapper-based systems.

Wrappers fit very well the Web data extraction problem because HTML pages, although lacking in semantic structure, are syntactically structured: HTML is just a presentation markup language, but wrappers can use tags to infer underlying information. Wrappers succeeded where IE NLP-based techniques failed: often Web pages do not own a rich grammar structure, so NLP cannot be applied with good results.

5.3 Semi-Automatic Wrapper Generation

5.3.1 *Visual Extraction.* Under this category we classify techniques that make it possible to the users to build wrappers from Web pages of interest using a GUI and interactively, without any deep understanding of the wrapper programming language, as wrappers are generated automatically by the system relying on users' directives.

5.3.1.1 *Regular-Expression-Based approach:.* One of the most common approaches is based on regular expressions, which are an old, but still powerful, formal language used to identify strings or patterns in unstructured text, defining match criteria. Rules could be complex so, writing them manually, could require too much time and a great expertise: wrappers based on regular expressions dynamically generate rules to extract desired data from Web pages. Usually writing regular expressions on HTML pages relies on the following criteria: word boundaries, HTML tags, tables structure, etc.

A notable tool implementing regular-expression-based extraction is W4F [Sahuguet and Azavant 1999], adopting an annotation approach: instead of putting users facing the HTML code, W4F eases the design of the wrapper through a wizard that allows users to select and annotate elements directly on the page; W4F builds the regular expression extraction rules of the annotated items and presents them to users, actually because it is not providing the best extraction rule, demanding to users the optimization step. W4F extraction rules, besides *match*, could also implement the *split* expression, which separates words, annotating different elements on the same string.

5.3.1.2 *Logic-Based approach:.* Tools based on this approach successfully build wrappers through a *wrapper programming language*, considering Web pages not as simply text strings but as pre-parsed trees, representing the DOM of the page. Gottlob and Koch [Gottlob and Koch 2004a] formalized the first wrapping language, suitable for being incorporated into visual tools, satisfying the condition that all its constructs can be implemented through *corresponding visual primitives*: starting from the unranked labeled tree representing the DOM of the Web page, the algorithm re-labels nodes, truncates the irrelevant ones, and finally returns a subset of original tree nodes, representing the selected data extracted. The extraction function of all these operations relies on monadic datalog [Gottlob and Koch 2004b]. They demonstrated that monadic datalog over tree structures is equivalent to monadic second order logic (MSO), and hence very expressive. However, unlike MSO, a wrapper in monadic datalog can be modelled nicely in a visual and interactive step-by-step manner.

Baumgartner et al. developed the Elog wrapping language as a possible implementation of a monadic datalog with minor restrictions, using it as the core extraction function of the Lixto Visual Wrapper [Baumgartner et al. 2001b; 2001a]: this tool provides a GUI to select, through visual specification, patterns in Web pages, in hierarchical order, highlighting elements of the document and specifying relationships among them; information identified in this way could be too general, so the system permits to *add some restricting conditions*, e.g. before/after, not-before/not-after, internal and range conditions. Finally, selected information are translated into XML using pattern names as XML element names, obtaining structured data from unstructured pages.

5.3.2 *Spatial Reasoning.* A completely different approach, called Visual Box Model, is using visual cues for understanding the presence of tabular data in HTML documents, not strictly represented under the `<table>` element [Krüpl et al. 2005; Gatterbauer and Bohunsky 2006]: the technique is based on the X-Y cut OCR algorithm, relying on the Gecko [28] rendering engine used by the Mozilla Web browser, to extract the CSS 2.0 visual box model, accessing the positional information through XPCOM. Cuts are recursively applied to the bitmap image (the rendering of the page) and stored into an X-Y tree, building a tree where ancestor nodes with leaves represent not-empty tables. Some secondary operations check that extracted tables contain useful information, because usually, although is a deprecated practice, many Web developers use tables for structural and graphical issues.

5.4 Automatic Wrapper Generation

By definition, the automatic wrapper generation implies no human interaction: techniques to recognize and extract relevant data were autonomously developed and systems with an advanced degree of automation and an high level of independent decision capability represent the state-of-the-art of the automatic wrapper generation approach.

5.4.1 *Automatic Matching.* RoadRunner [Crescenzi et al. 2001; Crescenzi and Mecca 2004] is an interesting example of automatic wrapper generator: this system is oriented to data-intensive websites based on templates or regular structures. This system tackles the problem of automatic matching bypassing common fea-

[28]https://developer.mozilla.org/en/Gecko

tures used by standard wrappers, typically using additional information, provided by users, labeling example pages, or by the system through automatic labeling, or *a priori* knowledge on the schema, e.g. on the page structure/template. In particular, RoadRunner relies on the fundamental idea of working with two HTML pages at a time in order to discover patterns while analyzing similarities and differences between structure and content of pages. Essentially RoadRunner can extract relevant information from any website containing at least two pages with similar structure: usually Web pages are dynamically generated and relevant data are positioned in the same area of the page, excluding small differences due, for example, to missing values. They called *class of pages* these kind of Web sources characterized by a common generation script. The problem is reduced to extracting the source data-set thus generating a wrapper starting from the inference of a common structure from the two-page-based comparison. This system can handle missing and optional values and also small structural differences, adapting very well to all kinds of data-intensive Web sources (e.g. based on templates), relying on a solid theoretical background, that ensures an high degree of precision of the matching technique.

5.4.2 *Partial Tree Alignment.* Zhai and Liu [Zhai and Liu 2005; 2006], theorized the partial tree alignment technique and developed a Web data extraction system based on it. This technique relies on the idea that information in Web documents usually are collected in contiguous regions of the page, called *data record regions*. Partial tree alignment consists in extracting these regions, e.g. using a tree matching algorithm, called *tree edit distance*. This approach works in two steps, segmentation and partial tree alignment: in the first phase the Web page is split in segments, without extracting data; this pre-processing is powerful because the system does not simply perform an analysis based on the DOM tree, but also relies on visual cues, like the spatial reasoning technique, trying to identify gaps between data records; it is useful also because helps the process of extracting structural information from the HTML tag tree, in that situations when the HTML syntax is abused, e.g. using tabular structure instead of CSS to arrange the graphical aspect of the page. After that, the partial tree alignment algorithm is applied to data records earlier identified: each data record is extracted from its DOM sub-tree position, constituting the root of a new single tree, because each data record could be contained in more than one non-contiguous sub-tree in the original tag tree. Partial tree alignment approach implies the alignment of data fields with certainty, excluding those which cannot be aligned, to ensure a high degree of precision; during this process no data items are involved, this because partial tree alignment works only on tree tags matching, represented as the minimum cost, in terms of operations (e.g. node removal, node insertion, node replacement), to transform one node in another one.

5.5 Wrapper Induction

Wrapper induction techniques differ from the latter essentially for the degree of automation: most of the wrapper induction systems need labeled examples provided during the training sessions, thus requiring human engagement; actually a couple of systems can obtain these information autonomously, representing, *de facto*, an hybrid approach between wrapper induction and automatic wrapper generation. In wrapper induction, extraction rules are learned during training sessions and then applied to extract data from Web pages similar to the example provided.

5.5.1 *Machine-Learning-Based approach.* Standard machine-learning-based techniques rely on training sessions to let the system acquire a domain expertise. Training a Web data extraction system, based on the machine-learning approach, requires a huge amount of labeled Web pages, before starting to work with an acceptable degree of precision. Manual labeling should provide both positive and negative examples, especially for different websites but also in the same website, in pages with different structure, because, usually, templates or patterns differ, and the machine-learning-based system have to learn how to extract information in these cases. Statistical machine-learling-based systems were developed relying on conditional models [Phan et al. 2005] or adaptive search [Turmo et al. 2006] as an alternative solution to human knowledge and interaction.

Many wrapper induction systems were developed relying on the machine-learning approach: Flesca et al. [Flesca et al. 2004] classified ShopBot, WIEN, SoftMealy, STALKER, RAPIER, SRV and WHISK [Soderland 1999], analyzing some particular features like support for HTML-documents, NLP, texts etc.

Kushmerick developed the first wrapper induction system, WIEN [Kushmerick 2000], based on a couple of brilliant *inductive learning techniques* that enable the system to automatically label training pages, representing, *de facto*, an hybrid approach to speed-up the learning process. Although these hybrid features, WIEN has many limitations, e.g. it cannot handle missing values.

SoftMealy, developed by Hsu and Dung [Hsu and Dung 1998], was the first wrapper induction system specifically designed for the Web data extraction: relying on a *non-deterministic finite state automata*, SoftMealy also uses a bottom-up inductive learning approach to extract wrapping rules. During the training session the system acquires training pages represented as an automaton on all the possible permutations of Web pages: states represent extracted data, while state transitions represent extraction rules. STALKER [Muslea et al. 1999] is a system for learning supervised wrappers with some affinity with SoftMealy but differing in the relevant data specification: a set of tokens is manually placed on the Web page identifying information that have to be extracted, ensuring the capability of handling empty values, hierarchical structures and unordered items. Bossa et al. [Bossa et al. 2006] developed a token based lightweight Web data extraction system called *Dynamo* that differs from STALKER because tokens are placed during the Web pages building to identify elements on the page and relevant information. This system is viable strictly in such situations when webmasters can modify page structures providing tokens placement to help the extraction system.

5.6 Wrapper Maintenance

Wrapper building, regardless the technique applied to generate it, is just an aspect of the problem of data extraction from Web sources: unlike static documents, Web pages dynamically change and evolve, and structure may change, sometimes with the consequence that wrappers cannot successfully extract data. Actually, a critical step of the Web data extraction process is the *wrapper maintenance*: this can be performed manually, updating or rewriting the wrapper each time Web pages change; this approach could fit well for small problems, but is not-trivial if the pool of Web pages is extended (e.g. a quite complex data extraction task include hundred thousand pages, usually dynamically generated and frequently updated). Kushmerick [Kushmerick 2002] defined the *wrapper verification* problem and, shortly, a couple of manual wrapper maintenance techniques were developed to handle simple problems. Ferrara and Baumgartner [Ferrara and Baumgartner 2010] developed a system of automatic wrapper adaptation relying on analyzing structural similarities between different version of the same Web page. We analyze a viable practice to automatically solve the problem of the wrapper maintenance.

5.6.1 *Schema-Guided Wrapper Maintenance.* Meng et al. [Meng et al. 2003] developed the SG-WRAM (Schema-Guided WRApper Maintenance) for Web data extraction starting from the observation that, changes in Web pages, even substantial, always preserve syntactic features (i.e. syntactic characteristics of data items like data patterns, string lengths, etc.), hyperlinks and annotations (e.g. descriptive information representing the semantic meaning of a piece of information in its context). They developed a Web data extraction system, providing schemes, starting from the wrapper generation, till to the wrapper maintenance: during the generation the user provides HTML documents and XML schemas, specifying mappings between them, later the system will generate extraction rules and then it will execute the wrapper to extract data, building an XML document with the specified XML schema; the wrapper maintainer checks extraction issues and provide an automatic repairing protocol for wrappers which fail the task because Web pages changed. The XML schema is in the format of a DTD (Document Type Definition) and the HTML document is represented as a DOM tree: SG-WRAM builds corresponding mappings between them and generates extraction rules in the format of a XQuery expression.

6. CONCLUSIONS

The World Wide Web contains a huge amount of unstructured data. The need for structured information urged researchers to develop and implement various strategies to accomplish the task of automatically extract data from Web sources. Their ultimate goal is to support business, social or commercial applications. In this paper we presented a survey on the problem of Web automatic data extraction, focusing on applications, approaches and techniques that were developed during last years. There are some future directions and challenges that can be foreseen. Some of them comprise how to address enormous scaling issues of the extraction problem, the robustness of the process and the design and implementation of auto-adaptive wrappers.

REFERENCES

Amalfitano, D., Fasolino, A. R., and Tramontana, P. 2008. Reverse engineering finite state machines from rich internet applications. In *WCRE '08: Proc. of the 2008 15th Working Conference on Reverse Engineering.* IEEE Computer Society, Washington, DC, USA, 69–73.

Anton, T. 2004. *XPath-Wrapper Induction by generalizing tree traversal patterns.* MIT Press, Cambridge, MA, USA.

Baumgartner, R., Campi, A., Gottlob, G., and Herzog, M. 2010. Web data extraction for service creation. *Search Computing: Challenges and Directions*.

Baumgartner, R., Ceresna, M., and Ledermuller, G. 2005. Deepweb navigation in web data extraction. In *CIMCA '05: Proc. of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*. IEEE Computer Society, Washington, DC, USA, 698–703.

Baumgartner, R., Flesca, S., and Gottlob, G. 2001a. The elog web extraction language. In *LPAR '01: Proc. of the Artificial Intelligence on Logic for Programming*. Springer-Verlag, London, UK, 548–560.

Baumgartner, R., Flesca, S., and Gottlob, G. 2001b. Visual web information extraction with lixto. In *VLDB '01: Proc. of the 27th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 119–128.

Baumgartner, R., Frölich, O., Gottlob, G., Harz, P., Herzog, M., Lehmann, P., and Wien, T. 2005. Web data extraction for business intelligence: the lixto approach. In *Proc. of BTW 2005*. 48–65.

Baumgartner, R., Fröschl, K., Hronsky, M., Pöttler, M., and Walchhofer, N. 2010. Semantic online tourism market monitoring. *Proc. of the 17th ENTER eTourism International Conference*.

Baumgartner, R., Gatterbauer, W., and Gottlob, G. 2009. Web data extraction system. *Encyclopedia of Database Systems*, 3465–3471.

Baumgartner, R., Gottlob, G., and Herzog, M. 2009. Scalable web data extraction for online market intelligence. *Proc. VLDB Endow. 2,* 2, 1512–1523.

Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist. 22,* 1, 39–71.

Berthold, M. and Hand, D. J. 1999. *Intelligent Data Analysis: An Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Boronat, X. 2008. A comparison of html-aware tools for web data extraction. M.S. thesis, Universität Leipzig, Fakultät für Mathematik und Informatik. Abteilung Datenbanken.

Bossa, S., Fiumara, G., and Provetti, A. 2006. A lightweight architecture for rss polling of arbitrary web sources. In *WOA*.

Chang, C.-H., Kayed, M., Girgis, M. R., and Shaalan, K. F. 2006. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng. 18,* 10, 1411–1428.

Chen, H., Chau, M., and Zeng, D. 2002. Ci spider: a tool for competitive intelligence on the web. *Decis. Support Syst. 34,* 1, 1–17.

Crescenzi, V. and Mecca, G. 2004. Automatic information extraction from large websites. *J. ACM 51,* 5, 731–779.

Crescenzi, V., Mecca, G., and Merialdo, P. 2001. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB '01: Proc. of the 27th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 109–118.

Dave, K., Lawrence, S., and Pennock, D. M. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW '03: Proc. of the 12th international conference on World Wide Web*. ACM, New York, NY, USA, 519–528.

Descher, M., Feilhauer, T., Ludescher, T., Masser, P., Wenzel, B., Brezany, P., Elsayed, I., Wöhrer, A., Tjoa, A. M., and Huemer, D. 2009. Position paper: Secure infrastructure for scientific data life cycle management. In *ARES*. 606–611.

Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., Lonsdale, D. W., Ng, Y.-K., and Smith, R. D. 1999. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowl. Eng. 31,* 3, 227–251.

Fayzrakhmanov, R., Goebel, M., Holzinger, W., Kruepl, B., Mager, A., and Baumgartner, R. 2010. Modelling web navigation with the user in mind. In *Proc. of the 7th International Cross-Disciplinary Conference on Web Accessibility.*

Ferrara, E. and Baumgartner, R. 2010. Automatic wrapper adaptation by tree edit distance matching. *Technical Report*.

FIUMARA, G. 2007. Automated information extraction from web sources: a survey. *Proc. of Between Ontologies and Folksonomies Workshop in 3rd International Conference on Communities and Technology*, 1–9.

FLESCA, S., MANCO, G., MASCIARI, E., RENDE, E., AND TAGARELLI, A. 2004. Web wrapper induction: a brief survey. *AI Commun. 17,* 2, 57–61.

GARRETT, J. J. 2005. Ajax: A new approach to web applications. Tech. rep., Adaptive Path.

GATTERBAUER, W. 2009. Web harvesting. *Encyclopedia of Database Systems*, 3472–3473.

GATTERBAUER, W. AND BOHUNSKY, P. 2006. Table extraction using spatial reasoning on the css2 visual box model. In *AAAI '06 Proc. of the 21st national conference on Artificial intelligence.* AAAI Press, 1313–1318.

GATTERBAUER, W., BOHUNSKY, P., HERZOG, M., KRÜPL, B., AND POLLAK, B. 2007. Towards domain-independent information extraction from web tables. In *WWW '07: Proc. of the 16th international conference on World Wide Web.* ACM, New York, NY, USA, 71–80.

GOTTLOB, G. AND KOCH, C. 2004a. Logic-based web information extraction. *SIGMOD Rec. 33,* 2, 87–94.

GOTTLOB, G. AND KOCH, C. 2004b. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM 51,* 1, 74–113.

HAMMERSLEY, B. 2005. *Developing feeds with rss and atom.* O'Reilly.

HAN, H. 2002. Conceptual modeling and ontology extraction for web information. Ph.D. thesis. Supervisor-Elmasri, Ramez.

HAN, J. AND KAMBER, M. 2000. *Data mining: concepts and techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

HSU, C.-N. AND DUNG, M.-T. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Inf. Syst. 23,* 9, 521–538.

HU, X., LIN, T. Y., SONG, I.-Y., LIN, X., YOO, I., LECHNER, M., AND SONG, M. 2004. Ontology-based scalable and portable information extraction system to extract biological knowledge from huge collection of biomedical web documents. In *WI '04: Proc. of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence.* IEEE Computer Society, Washington, DC, USA, 77–83.

IRMAK, U. AND SUEL, T. 2006. Interactive wrapper generation with minimal user effort. In *WWW '06: Proc. of the 15th international conference on World Wide Web.* ACM, New York, NY, USA, 553–563.

KAISER, K. AND MIKSCH, S. 2005. Information extraction. a survey. Tech. rep., E188 - Institut für Softwaretechnik und Interaktive Systeme; Technische Universität Wien.

KHARE, R. AND ÇELIK, T. 2006. Microformats: a pragmatic path to the semantic web. In *WWW '06: Proc. of the 15th international conference on World Wide Web.* ACM, New York, NY, USA, 865–866.

KRÜPL, B., HERZOG, M., AND GATTERBAUER, W. 2005. Using visual cues for extraction of tabular data from arbitrary html documents. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web.* ACM, New York, NY, USA, 1000–1001.

KUHLINS, S. AND TREDWELL, R. 2003. Toolkits for generating wrappers. In *NODe '02: Revised Papers from the International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World.* Springer-Verlag, London, UK, 184–198.

KUSHMERICK, N. 1997. Wrapper induction for information extraction. Ph.D. thesis, University of Washington. Chairperson-Weld, Daniel S.

KUSHMERICK, N. 2000. Wrapper induction: efficiency and expressiveness. *Artif. Intell. 118,* 1-2, 15–68.

KUSHMERICK, N. 2002. Finite-state approaches to web information extraction. *Proc. of 3rd Summer Convention on Information Extraction*, 77–91.

LAENDER, A. H. F., RIBEIRO-NETO, B. A., DA SILVA, A. S., AND TEIXEIRA, J. S. 2002. A brief survey of web data extraction tools. *SIGMOD Rec. 31,* 2, 84–93.

MANNING, C. D. AND SCHÜTZE, H. 1999. *Foundations of statistical natural language processing.* MIT Press, Cambridge, MA, USA.

Melomed, E., Gorbach, I., Berger, A., and Bateman, P. 2006. *Microsoft SQL Server 2005 Analysis Services (SQL Server Series)*. Sams, Indianapolis, IN, USA.

Meng, X., Hu, D., and Li, C. 2003. Schema-guided wrapper maintenance for web-data extraction. In *WIDM '03: Proc. of the 5th ACM international workshop on Web information and data management*. ACM, New York, NY, USA, 1–8.

Mika, P. 2007. Ontologies are us: A unified model of social networks and semantics. *Web Semant. 5,* 1, 5–15.

Milgram, S. 1967. The small world problem. *Psychology Today 2*, 60–67.

Monge, A. E. 2000. Matching algorithm within a duplicate detection system. *IEEE Techn. Bulletin Data Engineering 23,* 4.

Muslea, I., Minton, S., and Knoblock, C. 1999. A hierarchical approach to wrapper induction. In *AGENTS '99: Proc. of the third annual conference on Autonomous Agents*. ACM, New York, NY, USA, 190–197.

Phan, X., Horiguchi, S., and Ho, T. 2005. Automated data extraction from the web with conditional models. *Int. J. Bus. Intell. Data Min. 1,* 2, 194–209.

Plake, C., Schiemann, T., Pankalla, M., Hakenberg, J., and Leser, U. 2006. Alibaba: Pubmed as a graph. *Bioinformatics 22,* 19, 2444–2445.

Rahm, E. and Bernstein, P. A. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal 10,* 4, 334–350.

Rahm, E. and Do, H. H. 2000. Data cleaning: Problems and current approaches. *IEEE Bulletin on Data Engineering 23,* 4.

Sahuguet, A. and Azavant, F. 1999. Building light-weight wrappers for legacy web data-sources using w4f. In *VLDB '99: Proc. of the 25th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 738–741.

Sarawagi, S. 2008. Information extraction. *Found. Trends databases 1,* 3, 261–377.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Comput. Surv. 34,* 1, 1–47.

Soderland, S. 1999. Learning information extraction rules for semi-structured and free text. *Mach. Learn. 34,* 1-3, 233–272.

Tanaka, M. and Ishida, T. 2006. Ontology extraction from tables on the web. In *SAINT '06: Proc. of the International Symposium on Applications on Internet*. IEEE Computer Society, Washington, DC, USA, 284–290.

Turmo, J., Ageno, A., and Català, N. 2006. Adaptive information extraction. *ACM Comput. Surv. 38,* 2, 4.

Wang, P., Hawk, W. B., and Tenopir, C. 2000. Users' interaction with world wide web resources: an exploratory study using a holistic approach. *Inf. Process. Manage. 36*, 229–251.

Weikum, G. 2009. Harvesting, searching, and ranking knowledge on the web: invited talk. In *WSDM '09: Proc. of the Second ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, USA, 3–4.

Winograd, T. 1972. *Understanding Natural Language*. Academic Press, Inc., Orlando, FL, USA.

Zanasi, A. 1998. Competitive intelligence through data mining public sources. In *Competitive intelligence review*. Vol. 9. Wiley, New York, NY, ETATS-UNIS (1990-2001) (Revue), 44–54.

Zhai, Y. and Liu, B. 2005. Web data extraction based on partial tree alignment. In *WWW '05: Proc. of the 14th international conference on World Wide Web*. ACM, NY, 76–85.

Zhai, Y. and Liu, B. 2006. Structured data extraction from the web based on partial tree alignment. *IEEE Trans. on Knowl. and Data Eng. 18,* 12, 1614–1628.

Zhao, H. 2007. Automatic wrapper generation for the extraction of search result records from search engines. Ph.D. thesis, State University of New York at Binghamton, Binghamton, NY, USA. Adviser-Meng, Weiyi.

Received; revised;