A Survey on Region Extractors from Web Documents

Hassan A. Sleiman and Rafael Corchuelo

Abstract—Extracting information from web documents has become a research area in which new proposals sprout out year after year. This has motivated several researchers to work on surveys that attempt to provide an overall picture of the many existing proposals. Unfortunately, none of these surveys provide a complete picture, because they do not take region extractors into account. These tools are kind of preprocessors, because they help information extractors focus on the regions of a web document that contain relevant information. With the increasing complexity of web documents, region extractors are becoming a must to extract information from many websites. Beyond information extraction, region extractors have also found their way into information retrieval, focused web crawling, topic distillation, adaptive content delivery, mashups, and metasearch engines. In this paper, we survey the existing proposals regarding region extractors and compare them side by side.

Index Terms—Information extractors, wrappers, web documents, region extractors, enterprise information integration

1 Introduction

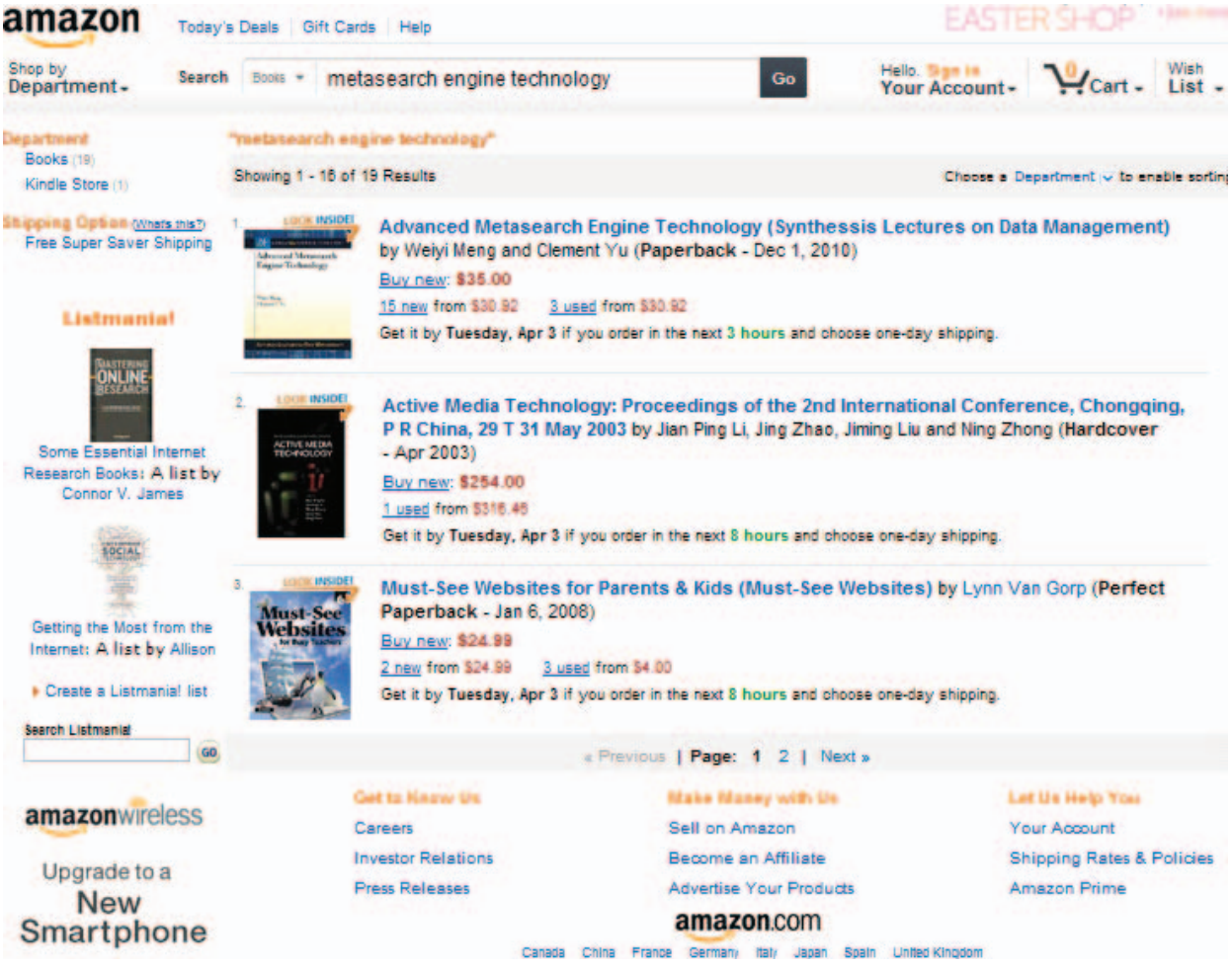
The majority of companies world-wide use the web to provide their customers with catalogues of products and/or services, not to mention news spots, blogging sites, social networks, virtual libraries, and so on. This turns the web into the largest repository of information in human-friendly formats. Unfortunately, using this information in automated business processes is not easy at all because it is buried into text and/or formatting tags.

This has motivated many researchers to work on proposals to analyze web documents and extract their information in structured formats automatically; these proposals are commonly referred to as information extractors or wrappers [9], [35], [91], [92], [95], [113], [121], [137], [157]. Web information extraction is the task of identifying, extracting, and structuring relevant information from web documents in structured formats, e.g., tables or XML. (Note that relevancy depends completely on the context.) For instance, Fig. 1 shows a sample web document from Amazon to which we have applied DeLa [164], which is a well-known information extractor. This information extractor identifies the data region in the web document using a region extractor called DSE [163], searches for repeating patterns inside this region to separate the data records, aligns these records to extract their attributes, and then, labels and formats these attributes in a results table.

Information extractors can be broadly classified according to whether they deal with free-text web documents or semistructured web documents. The relevant information in a free-text web document is buried into sentences that are written in (telegraphic) natural language, whereas in semistructured web documents it is buried into HTML scripts. Note that a free-text web document usually contains HTML tags that provide a little structure, e.g., <h1> tags to typeset a title, a to typeset the authors, <p> tags to typeset paragraphs, and or <emph> tags to emphasize a piece of text; unfortunately, this structure is far too light to be useful for information extraction purposes. Free-text web documents require natural language processing techniques to extract information from them. Contrarily, semistructured web documents generally use HTML tags to typeset small pieces of information (attributes) in tables or lists. Independently from the kind of document on which it can work, the goal of an information extractor is to identify pieces of relevant information within a document and return them in a structured format.

The literature provides a collection of surveys on information extraction. Turmo et al. [157] and Sarawagi [137] provided two surveys regarding the many existing techniques to extract information from free-text web documents; their conclusion was that roughly half the proposals are based on rules [3], [15], [30], [60], [77], [86], [133], [146], [147], [156], [174] and the other half are based on statistical models [37], [38], [43], [57], [62], [63], [83], [118], [128], [135], [138], [144], [150], [177], [181]. Chang et al. [35] provided the most recent survey regarding proposals for information extraction from semistructured web documents. Their focus was on information extractors that rely on so-called extraction rules, which can be either learned from examples [8], [13], [26], [27], [33], [34], [41], [45], [59], [66], [67], [73], [74], [75], [85], [89], [93], [99], [103], [104], [122], [126], [147], [151], [164], [178], [182] or handcrafted [10], [44], [64], [70], [116], [130], [136]. Meng and Yu [113] surveyed information extraction techniques that can be used to extract search result records returned by search engines. These surveys did not take into account either some proposals that build on heuristics, i.e., built-in extraction rules that have proven to work well in many common cases [6], [51], [68], [98], [107], [143], [149] or workbenches, which are targeted toward end

• *The authors are with the University of Sevilla, ETSI Informática, Avda. Reina Mercedes, s/n, Sevilla E-41012, Spain.*
E-mail: {hassansleiman, corchu}@us.es.



Title	Authors	Date	Price
Advanced Metasearch Engine Technology (Synthessis Lectures on Data Management)	Weiyi Meng and Clement Yu	Dec 1, 2010	$35.00
Active Media Technology: Proceedings of the 2nd International Conference, Chongqing, P R China, 29 T 31 May 2003	Jian Ping Li, Jing Zhao, Jiming Liu and Ning Zhong	Apr 2003	$254.00
Must-See Websites for Parents & Kids (Must-See Websites)	Lynn Van Gorp	Jan 6, 2008	$24.99

Fig. 1. Extracting information from a semistructured web document using DeLa.

users because they facilitate putting other proposals into practice in production scenarios [1], [16], [49], [56], [71], [94], [102], [105], [119], [120]. Unfortunately, none of the previous proposals is universally applicable, which makes information extraction quite an active research area [48].

We define a region in a web document as an HTML fragment that shows information about one or more related items when it is rendered on a web browser. Such items can be data records, e.g., information about products, services, goods, or pieces of news, headers with navigation menus, footers with contact and corporate information, or sidebars with advertisements, to mention a few examples. In the sequel, we make a distinction between individual data records, data regions (which encompass a series of data records), and the rest of regions, to which we refer to as ancillary regions. The majority of region extractors focus on data records and data regions. (Note that it is not unusual to find several data regions in the same web document.) For instance, Fig. 2 illustrates the regions identified by VIPS [24] in our running example. It has identified a total of 15 regions, which we classify into: one data region that contains three data records and 11 ancillary regions around the data region. Note that VIPS only identifies and separates regions in a web document; it is the user's responsibility to select the region(s) of interest.

Typical information extraction tasks focus on data regions and data records. That implies that as the complexity of typical web documents increases, information extractors have to analyze more and more irrelevant regions, which has an impact on both efficiency and effectiveness [84], [163], [175]. This has motivated a number of authors to work on region extractors as a means to relieve information extractors from the burden of analyzing

Fig. 2. Regions identified by VIPS.

many regions of a web document that do not contain any relevant information [19], [23], [24], [53], [84], [97], [100], [114], [125], [141], [163], [169], [179], [180]. The difference between information extractors and region extractors is that the former focus on extracting and structuring data records and their attributes, whereas the latter focus on identifying the HTML fragments that contain this information. Yi et al. [175], Wang and Lochovsky [163], and Kang and Choi [84] have confirmed experimentally that using region extractors has a positive impact on both efficiency and effectiveness; it is not surprising then that some recent proposals for information extraction incorporate a built-in region extractor [98], [99], [143], [164], [178]. Beyond information extraction, region extractors have also proven to be useful for information retrieval [176], focused web crawling [25], [32], topic distillation [31], adaptive content delivery [82], mashups [152], and metasearch engines [113].

The literature records an increasing number of proposals in this area [19], [23], [24], [53], [84], [97], [100], [114], [125], [141], [163], [169], [179], [180]. Unfortunately, none of the surveys regarding information extraction that we have found in the literature take them into account [35], [91], [92], [95], [121], [137], [157]. The only exception is the survey by [113], who studied three techniques that are specifically tailored toward search engines or that can be adapted for this purpose, namely ViDRE [169], ViNTS [179], and OMINI [23]. This has motivated us to work on this paper to complete the overall picture regarding information extraction. We have surveyed all of the proposals on region extractors of which we are aware within a four-dimensional comparison framework in which we have compared them side by side. Our main conclusions are the following:

1. more effort is required regarding free-text web documents because the proposals we have surveyed rely almost exclusively on finding regular structures in a web document;
2. the majority of proposals seem scalable because they are unsupervised;
3. there are no conclusive and comparable results regarding their efficiency and effectiveness; and
4. none of them is universally applicable.

The rest of the paper is organized as follows: In Section 2, we report on previous surveys of the literature and describe our comparison framework briefly; In Section 3, we report on current state-of-the-art region extractors; then, we compare them in Section 4; we present our conclusions in Section 5. The paper finishes with a list of references to selected articles.

2 Related Work

The large number of existing proposals on information extraction has motivated several authors to work on surveys

TABLE 1
Comparison between Previous Surveys that Just Provide a Taxonomy

Survey	Focus	Taxonomy	Techniques
Hsu and Dung [74] (1998)	Information extractors for free-text and semi-structured documents.	1) Hand-crafted proposals using general-purpose programming languages; 2) Specific-purpose languages and tools; 3) Heuristic-based proposals; 4) Extraction rule learners.	TSIMMIS [69], InfoExtractor [145], EDITOR [13], ShopBot [49], Ashish and Knoblock [11], FAQ Miner [79], WIEN [92], STALKER [122].
Muslea [121] (1999)	Information extractors for free-text and semi-structured documents.	1) Information extractors based on lexical/syntactic/semantic features; 2) Information extractors based on delimiters; 3) Hybrid information extractors that use both lexical/syntactic/semantic features and text delimiters.	AutoSlog [132], LIEP [76], PALKA [85], CRYSTAL [148], Webfoot [146], HASTEN [89], WHISK [147], RAPIER [25], SRV [58], WIEN [92], SoftMealy [74], STALKER [122].
Kushmerick and Thomas [91] (2003)	Information extractors for free-text and semi-structured documents.	1) Finite-state proposals based on automata; 2) Relational proposals based on first-order logic.	WIEN [92], STALKER [122], SoftMealy [74], BWI [60], WHISK [147], Freitag and McCallum [61], Brin [20], RoadRunner [44], Autoslog [132], T-Wrapper [155], Ciravegna [39], Grieser and others [64], Junker and others [80], SRV [58], CRYSTAL [148], RAPIER [25], MIA [16].
Sarawagi [137] (2007)	Information extractors for free-text documents.	1) Input features; 2) Output structures; 3) Persistence features; 4) Extraction method; 5) Resources availability.	FASTUS [7], CIRCUS [95], AutoSlog [132], Aitken [4], RAPIER [25], WHISK [147], Avatar [78], DBLife [139], SRV [58], CRAM [2], LP2 [39], Nymble [17], DATAMOLD [19], Seymore and others [138], Klein and Manning [87], Malouf [108], Freitag and Mccallum [62], Ratnaparkhi [131], Shilman and others [142], Viola and Narasimhan [160], Gate [46], UIME [54], Maynard and others [109], Bunescu and others [21], MALLET [110], Wu and Weld [169], Choi and others [38], Cumby and Roth [45], TEG [53], Ramakrishnan and others [129].
Meng and Yu [113] (2010)	Information extractors specifically tailored towards search engines or information extractors that can be adapted for this purpose.	1) Supervised information extraction techniques; 2) Unsupervised information extraction techniques: 2.1) Techniques based on tag information; 2.2) Techniques based on visual information; 2.3) Techniques based on tag and visual information; 2.4) Techniques based on tag, visual and domain information.	WIEN [92], SoftMealy [74], STALKER [122], Lixto Wrapper [15], Thresher [72], RoadRunner [44], OMINI [22], DeLa [164], EXALG [8], ViNTs [178], ViPER [143], DEPTA [177], ViDRE [102], ODE [149].

of the literature that provide comparison frameworks other researchers can use to have an overall picture of what the latest achievements are and how they compare to each other. By comparison framework, we mean a taxonomy to classify the existing proposals into several categories and/or a set of comparison features that allow to compare them side by side; related features can be grouped into independent dimensions; ideally, the majority of features should be objective, but we have found a few subjective features in the literature (see below).

The surveys in the literature can be classified into two groups according to how they compare the proposals. The first group includes the surveys that just provide a taxonomy, cf. Table 1, whereas the latter group includes surveys that also present a comparison framework, study the pros and cons, and compare the techniques side-by-side, cf. Tables 2 and 3.

Regarding the surveys that just provide a taxonomy, our conclusions are as follows:

- To the best of our knowledge, Hsu and Dung [75] were the first to provide a taxonomy according to which the existing proposals were classified into four categories, namely: Hand-crafted information extractors using general-purpose programming languages, specific-purpose languages and tools, heuristic-based proposals, and extraction rule learners.
- Muslea [121] presented a new taxonomy that separated information extraction techniques into three categories, namely: Information extractors that are based on lexical/syntactic/semantic features, information extractors that are solely based on delimiters around the pieces of text to be extracted, and hybrid information extractors that use both lexical/syntactic/semantic features and text delimiters.

TABLE 2
Comparison between Previous Surveys that Provide a Comparison Framework (1/2)

Survey	Focus	Taxonomy	Comparison features	Techniques
Laender and others [94] (2002)	Information extractors for free-text and semi-structured documents.	1) Information extraction languages; 2) HTML-based proposals; 3) Natural language proposals; 4) Induction algorithms to learn extraction rules; 5) Proposals that map text fragments onto user-defined data structures; 6) ontology-based proposals.	1) Degree of automation; 2) Support to extract hierarchical records; 3) Whether they deal with semi-structured documents or not; 4) How easy it is to use them; 5) Whether they produce XML output or not; 6) Whether they can deal with non-HTML sources; 7) How resilient and adaptive they are.	Minerva [43], TSIMMIS [69], WebOQL [9], W4F [136], XWRAP [101], RoadRunner [44], WHISK [147], RAPIER [25], SRV [58], WIEN [92], SoftMealy [74], STALKER [122], NoDoSE [1], DEByE [93], BYU [51].
Kuhlins and Tredwell [90] (2002)	Toolkits to extract information for free-text and semi-structured documents	1) Non-commercial toolkits; 2) Commercial toolkits.	1) Non-commercial toolkits: 1.1) Output data; 1.2) API; 1.3) Open source; 1.4) Web crawling; 1.5) GUI; 1.6) Scripting language; 1.7) Tool support; 1.8) Source code;. 2) Commercial toolkits: 2.1) company; 2.2) Demo version; 2.3) Output data; 2.4) Database connectivity; 2.5) API; 2.6) Web crawling; 2.7) GUI; 2.8) Editor; 2.9) Scripting language.	Araneus [111], BYU [51], DEByE [93], Jedi [75], LAPIS [117], RoadRunner [44], Scout [5], SoftMealy [74], TSIMMIS [69], WebL [86], Web Sphinx [115], WIEN [92], XWRAP [101], WebQL [168], vTag [162], TextPipe [153], Content Extractor [41], Unwwwrap [159], Agent Builder [55], FirstRain Studio [57], Intelligent Miner [77], ParserStudio [127], RoboSuite [134], Web Activity [165], Visual Web Task [161], Lixto Wrapper [15], WebDataKit [166], UnoStudio [158], X-Fetch Wrapper [170], Online Miner [124], Webinator [167], W4F [136], XRover [171].

- Kushmerick and Thomas [92] presented a survey in which they provided the following two-category taxonomy: Finite-state proposals that are based on automata and relational proposals that are based on first-order logic.
- Sarawagi [137] provided an additional survey on proposals to deal with free-text documents. The survey presented a three-category taxonomy based on the type of the technique used to extract information from free-text documents, namely: Techniques based on rule-based methods, techniques based on statistical models, and techniques based on both rules and statistical models.
- Meng and Yu [113] presented a survey on proposals that can be used to extract search result records from web documents returned by search engines. They created a taxonomy of two categories, namely: Supervised information extraction techniques and unsupervised information extraction techniques. The techniques in the latter category were classified into four subcategories, namely: Techniques based on tag information, techniques based on visual information, techniques based on tag and visual information, and techniques based on tag, visual and domain information. This survey includes three region extractors [23], [179], [169].

Regarding the surveys that provide a taxonomy and a set of comparison features, our conclusions are as follows:

- Laender et al. [95] presented a taxonomy that distinguishes among six categories, namely: Information extraction languages, HTML-based proposals, natural language proposals, induction algorithms to learn extraction rules, proposals that map text fragments into user-defined data structures, and ontology-based proposals. The comparison features included seven features, namely: Degree of automation, support to extract hierarchical records, whether they deal with semistructured documents or not, how easy it is to use them, whether they produce XML output or not, whether they can deal with non-HTML sources, and how resilient and adaptive they are.
- Kuhlins and Tredwell [91] a taxonomy that classified information extraction toolkits into two categories, namely: Noncommercial and commercial. In the case of noncommercial toolkits, they compared the following features: Whether they output XML or text, whether they provide an API or not, whether they are open-source or not, whether they support web crawling or not, whether they have an end-user interface or not, whether they provide an editor or not, whether they provide a scripting language or not, whether these toolkits provide support to end

TABLE 3
Comparison between Previous Surveys that Provide a Comparison Framework (2/2)

Survey	Focus	Taxonomy	Comparison features	Techniques
Turmo and others [156] (2006)	Information extractors for free-text documents.	1) Proposals based on extraction rules; 2) Proposals based on statistical models.	1) The learning paradigm; 2) The learing strategy; 3) Whether the technique learns knowledge that is useful to extract text fragments including its context or not; 4) Features of the input documents.	AutoSlog [132],AutoSlog-TS [133], PALKA [85], Chai and Biermann [28], TIMES [29], Basili and others [14], CRYSTAL [146], WAVE Aseltine [10], ExDISCO [173], Essence [27], DIPRE [20], Snowball [3], LIEP [76], WHISK [147], EVIUS [157], SRV [59], RAPIER [25], Seymore and others [138], Freitag and McCallum [61], Freitag and Mccallum [62], Peshkin and Pfeffer [128], Skounakis and others [144], Miller and others [118], Chieu and Ng [36], Cox and others [42], Kambhatla [82], Sun and others [150], Alice-SVM [37], Zelenko and others [176], Zhao and Grishman [180], SNoW-IE [135], ELIE [56].
Chang and others [32] (2006)	Information extractors for semi-structured documents.	1) Handcrafted; 2) Supervised; 3) Semi-supervised; 4) Unsupervised	1) Task domains: 1.1) Document type; 1.2) non-HTML support; 1.3) Extraction level; 1.4) Extraction targets variation; 1.5) Template variation; 1.6) Un-tokenised attributes; 2) Techniques used: 2.1) Scan pass 2.2) Extraction rules type; 2.3) Features used; 2.4) Learning algorithm; 2.5)Tokenisation schema; 3) Automation degree: 3.1) User expertise; 3.2) Fetch support; 3.3)Output/api support; 3.4) Applicability; 3.5) Limitations.	Minerva [43], TSIMMIS [69], WebOQL [9], W4F [136], XWRAP [101], RAPIER [25], SRV [58], WHISK [147], NoDoSE [1], DEByE [93], WIEN [92], STALKER [122], SoftMealy [74], IEPAD [34], OLERA [33], DeLa [164], RoadRunner [44], EXALG [8], DEPTA [177].

users or not, and their source code language. In the case of the commercial toolkits, they compared their vendors, whether they provide a demo version or not, the output, whether they allow connectivity to a database or not, whether they provide an API or not, whether they allow web crawling or not, whether they provide a user interface or not, whether they provide an editor or not, and whether they provide a scripting language or not. However, none of their comparison features provides an insight into the techniques behind the scenes since they all focus on how the toolkits can be used.

- Turmo et al. [157] created a taxonomy in which they classified machine learning techniques used for information extraction into two categories, namely: Techniques that learn rules and technique that learn statistical models. Then, they compared the following features for each category: The learning paradigm, the learning strategy, whether the technique learns knowledge that is useful to extract text fragments including its context or not, and features of the input documents. The authors compared fifteen free-text information extraction systems using the following features: Syntactic parsing, semantic interpretation, and discourse analysis.
- Chang et al. [35] presented an additional survey in which they classified the existing proposals into four categories according to their degree of user-supervision, namely: Handcrafted, supervised, semi-supervised, and unsupervised proposals. They devised a three-dimensional framework to compare them, namely: Features regarding how the proposal can adapt to formatting variations, aka. task domains, features regarding the technique used, and features regarding the degree of user intervention required.

The previous surveys provide a partial view of the whole range of proposals in the area of information extraction because only one of them reports on three region extractors and none of them provides a comparison framework that is specifically tailored to region extractors. This has motivated us to work on this survey. We have created a four-dimensional comparison framework in which each dimension reports on several related objective features. The first dimension includes a collection of features that are related to input requirements and output characteristics; the second

dimension focuses on features that are related to the algorithms used in each proposal; the third dimension provides several features that are related to performance; the last dimension is about miscellaneous features. All of the features on which we report are objective and can be contrasted in the literature.

3 State of the Art

In the following sections, we summarize the current state-of-the-art proposals regarding region extraction from web documents. In each case, we report on the most up-to-date references to the literature, what they are intended for, provide a synopsis of the hypothesis on which they rely, known problems, and a short description of the techniques behind the scenes; we also mention if the proposal being examined has inspired or put the foundation to other proposals.

3.1 Embley et al.

The proposal by Embley et al. [53] is intended to extract the data records from the largest data region in a web document. It builds on the hypothesis that there is a unique data region, which is the largest region in the web document, that this region contains multiple data records, that some tags are more likely to be data record separators based on their type and their occurrences, and that counting on an ontology helps identify data records. Some of the heuristics proposed by Embley et al. were used in the region extractor OMINI [23].

The algorithm works as follows:

1. It converts the HTML code of the input document into XHTML and removes comment tags. Then, it builds the DOM tree.
2. It detects the root node of the largest data region by searching for a so-called highest fan-out node in the DOM tree, i.e., the node with the highest number of immediate child nodes.
3. The algorithm calculates the occurrences of immediate child nodes in the highest fan-out node and considers the nodes whose frequency is less than 10 percent as separator candidates for the data records. If only one separator candidate is found, then it is considered as the data record separator in the data region. Otherwise, some heuristics are applied to the highest fan-out node contents and combined to detect the separators. These heuristics are: The tags that appear many times are more likely to be separating tags when the data region contains many data records, some types of tags have more probability to be separating tags based on typical web development practices, a tag is more likely to be a separator if the standard deviation of the size of the information between the occurrences of this tag is low, and a repeating tag sequence is more likely to be a separator. The technique also relies on a user-defined ontology that provides information about so-called record-identifying fields, i.e., sort of keywords that help identify data records and separate them.
4. The algorithm applies Stanford's certainty theory [106] to combine the previous heuristics and identify the tags that better separate the data records, which are returned.

Note that this proposal needs an user-defined ontology, but it is considered unsupervised because it is not mandatory according to the authors; contrarily, Buttler et al. [23] have found that this ontology plays a critical role in achieving high accuracy.

3.2 OMINI

OMINI [23] is intended to learn rules to extract data records from web documents that contain multiple data records in a unique data region. It builds on the hypothesis that there is a unique data region in the web document, that this region corresponds to the subtree with the largest number of children, and that some tags are more likely to be data record separators based on their occurrences and type. OMINI is a constituent part of the information extraction toolkit called XWRAP Elite [71].

The OMINI algorithm works as follows:

1. The DOM tree of the input document is built.
2. The algorithm searches for the data region using the following heuristics: It is the subtree with the largest number of children, it is has the largest contents (measured in bytes), and the largest number of tags.
3. It then works on detecting the data record separators inside the identified data region. It considers the child tags of the data region subtree node as record candidate separator tags, and ranks them according to five heuristics, namely: A tag is more likely to be a separator if the standard deviation of the size of the information between the occurrences of this tag is low, pairs of tags that appear several times consecutively without any text between them are more likely to be separators, some types of tags have more probability to be separating tags based on typical web development practices, pairs of tags that are immediate siblings and that appear several times are more likely to separate data records because their number of occurrences is the same as the number of data records in the data region, and the record candidate separator whose path to the other nodes in the data region has many occurrences is more likely to be a record separator since data records usually have a similar structure.
4. It combines the results from the heuristics using an approach for combining evidences from two or more independent observations [72]. A filtering step is then applied to the extracted data records to discard those that are not similar to the majority in terms of their tags and sizes. The idea is to discard false data records in the data region, e.g., advertisements.

Note that OMINI learns extraction rules, but the authors did not provide too much insight into their structure. However, according to [113], the rules include the DOM path to the root of the data region subtree, and a list of separator tags identified by OMINI to separate this data region into data records.

3.3 MDR: Mining Data Records

Mining Data Records [100], or MDR for short, is intended to extract data records. It builds on the hypothesis that a data region contains a repetitive structure in a document, that each repetitive structure inside a data region is a data record, and that data records are usually rendered inside tables and forms. Some recent information extractors have got inspiration from MDR, including [178], [99], and [143].

The MDR algorithm works as follows:

1. The DOM tree of the input document is built.
2. It then uses a combinatorial algorithm to find so-called generalized nodes, which are subsets of nodes that fulfill the following conditions: They are siblings, they are adjacent, they have the same number of children, and the edit distance among them does not exceed a predefined threshold. The idea is to detect regions that contain repetitive similar structures. The edit distance is calculated on the strings that result from serializing the nodes to be compared as strings; this serialization does not take text nodes into account, only HTML tags.
3. The subsets of generalized nodes that result from the previous step are considered as data regions because each data region is supposed to contain two or more data records that have similar structures.
4. The algorithm then separates the data records inside the previous data regions using the following heuristics:
    a. if the region consists of only one generalized node, it then checks if this node is not a table row, but all of its children are similar; if the condition is met, then the children are returned as independent data records; otherwise the generalized node itself is returned as a data record.
    b. If the generalized node contains two or more nodes with the same number of children and these children are similar to each other, then it means that they are noncontiguous data records, i.e., the data region is an HTML table in which each data record is formatted in columns and not in rows; otherwise, the whole generalized node is returned.

3.4 TPC: Tag Path Clustering

Tag Path Clustering [114], or TPC for short, is a proposal that is intended to extract all of the data records from all of the data regions in a web document. It is based on the hypothesis that a data region contains multiple contiguous or noncontiguous data records, that they are rendered similarly, visually aligned, and contain at least three HTML tags.

The TPC algorithm works as follows:

1. The DOM tree is first built, and the DOM paths of every node are calculated.
2. The algorithm works on mining visually repeating information using the DOM paths. For this purpose, it works with the HTML code as an ordered sequence of opening HTML tags of size n. For each DOM path p, it calculates a so-called visual signal vector, which is an occurrence vector for each possible DOM path (without repetition). The visual vector for a DOM path p is a binary vector of size n; it contains a 1 at position i if the DOM path for the tag at position i in the sequence of opening tags is the same as p, and 0 in other case.
3. The authors defined a visual signal as a triple (p, s, O), where p is a DOM path, s is its visual signal vector, and O is a collection that contains the individual occurrences, i.e., the nodes whose DOM path is p.
4. The algorithm works on the collection of visual signals. It builds a similarity matrix between the visual signals using a similarity function that relies on signal theory. Then, the similarity matrix is fed into a spectral clustering technique [123] that groups similar visual signals. The idea is that each cluster contains visual signals that belong to the same data region.
5. The algorithm discards clusters that contain less than three visual signals because it considers that a data record must contain at least three HTML tags. It also considers that the visual signals that belong to the same cluster represent the same data region.
6. The authors then introduce ancestor and descendant relationships between visual signals in each cluster, and determine the visual signals that are the maximal ancestors, i.e., visual signals that do not have any ancestors in their cluster.
7. If there is a single maximal ancestor in a cluster, then the algorithm considers that some of the nodes in the occurrence collection in the maximal ancestor are data records and that some of these nodes may contain multiple data records. It iteratively performs two steps until finding the best data record separators; first, it detects data record candidates by selecting node occurrences that have many descendants in the cluster, and then searches for a separator between the data record candidates by applying a technique that is similar to MDR [100], but uses the width and the height of the visual signal occurrence collection instead of the tree-edit distance. If there are multiple maximal ancestors, then it considers they are consecutive siblings that represent a data record, and the algorithm tries to detect a repeating pattern from a sequence of occurrences in different signals. It applies a heuristic in which the occurrences in the first maximal ancestor are considered record boundaries, creates data records candidates, and applies the steps described in the first case.
8. The algorithm tries to detect nested data records by applying the following heuristic: If a visual signal occurs at each point where data records are separated, then this visual signal corresponds to a visual pattern that separates two data records, and the text contained in this visual signal describes the relationship between these data records.

3.5 DSE: Data-Rich Section Extraction

Data-rich Section Extraction [163], or DSE for short, is intended to extract data regions. It builds on the hypothesis that documents from the same website have a common

layout that includes ancillary regions such as headers, footers, menus, or banners, whose structure is identical from document to document; what differentiates them is the structure of the data region, which should be similar only in documents that come from the same section. This proposal is a constituent part of a well-known information extractor called DeLa [164].

The DSE algorithm works as follows:

1. The user provides an input document. The algorithm then selects a document that can be reached from this document using a similarity measure that is based on their URLs. Both documents are then transformed into DOM trees.
2. The algorithm now searches for matching nodes in the previous trees. The matching is based on a recursive similarity function that takes the tag, a limited subset of attributes, and the number of children of every node into account. The idea is that similar nodes in different web documents constitute ancillary regions.
3. The nodes that are similar are removed from the original DOM trees, which results in two pruned trees that are likely to contain data regions. The data region that is selected by DSE is the pruned tree that corresponds to the input document.

3.6 U-REST: Unsupervised Record Extraction SysTem

Unsupervised Record Extraction SysTem [140], [141], or U-REST for short, is intended to extract data records. It builds on the hypothesis that data records in a document belong to a unique data region, have similar DOM trees, have similar structure, and have small separators, if any.

The U-REST algorithm works as follows:

1. The DOM tree of the input document is built and all of its subtrees are stored in a collection. Subtrees that have tags that cannot clearly represent a data record are removed. This includes subtrees with tags html, head, title, or script to mention a few.
2. The algorithm now clusters the remaining subtrees using a similarity function that is based on three features, namely: The tree-edit distance between subtrees, the string edit distance between their DOM paths, and a so-called trigram model, which is a vector that contains the HTML tags of the previous, next, and the root nodes of each subtree. The similarity function must be learned from samples.
3. U-REST now ranks the clusters according to a scoring function that must also be learned from samples. This scoring function is based on the following features: A contiguity function that measures the amount of text between contiguous subtrees, a content-coverage function that measures the amount of contents provided by the subtrees in a cluster with regard to the total amount of contents in the input document, and a variation function that measures both the internal irregularity of the subtrees in a cluster and their external regularity.
4. The algorithm now selects the cluster with the highest score as the data region because it contains similar subtrees that are supposed to be the data records. The subtrees in this region are returned as the data records in the input document.

Note that U-REST relies on two functions that must be learned from sample documents using the SVM method. However, the method is considered unsupervised because once these functions are learned, they can be reused with a variety of documents; in other words, the proposal does not require to be trained for specific websites.

3.7 STAVIES

STAVIES [125] is intended to extract data records. It builds on the hypothesis that relevant information resides in leaf nodes in the DOM tree, attributes inside data records share a high number of ancestors, at least 20 percent of the data in a document are relevant, and that data records have a repetitive structure.

The STAVIES algorithm works as follows:

1. The HTML code of the input document is converted into XHTML, its DOM tree is built, and the leaf nodes are stored in a pool.
2. The first step attempts to locate the data region within the input document. To do so, STAVIES relies on an algorithm that clusters the leaf nodes hierarchically. The similarity between every two nodes is calculated as the number of common ancestors, and two adjacent nodes are put in the same cluster at level l as long as the similarity between the child nodes is greater than or equal to l.
3. The algorithm returns the cluster that has less statistical variance and contains at least 20 percent of the total number of leaf nodes as the data region.
4. The authors realized experimentally that if leaf nodes are plotted against similarity, the result is quite a regular periodic signal. They developed a method to calculate a threshold that is based on calculating the semiperiod of the previous signal. This threshold is then used to cluster the nodes in the data region again. The authors used an outlier detection method for signals to remove noisy clusters.
5. STAVIES returns the clusters it identifies within the data region as data records.

3.8 VIPS: Vision-Based Page Segmentation

VIsion-based Page Segmentation [24], or VIPS for short, is intended to find all of the regions of which a document is composed. It builds on the hypothesis that web designers provide visual cues that help people recognize the different regions of which a document is composed, e.g., horizontal or vertical rules, boxes, colored panels, special fonts, or background images. Proposals like ViDRE [169], RIPB [84], and VSDR [97] endow VIPS with an algorithm to determine which the data regions are.

The VIPS algorithm works as follows:

1. The DOM tree of the input document is built, and it is enriched with information about visual features, e.g., position, background color, foreground color, font information, or background image.
2. Initially, the algorithm assumes that the whole document is a big region; it then traverses the DOM tree level after level and analyzes each node to

determine if it can be considered a subregion. To do so, the authors devised a collection of 12 heuristics, including the following: If a parent node has a child node of type hr, then that node must be divided into two subregions; if the background color or a node is different from the background color of one of its children, then that child is a subregion; if a table cell does not have any subregions, then the next table cell should not have any subregions, either; and so on.

3. After discovering the subregions in each level of the DOM tree, the algorithm calculates a set of separators, which are visual boxes that do not intersect any of the subregions. In other words, a separator is an empty region within a document. Each separator is assigned a weight that is related to the visual difference between the regions that it separates. The authors devised a number of heuristic to calculate the weight of a separator building on how similar the blocks it separates are, what colors they have, if a horizontal rule overlaps the separator, and so on. Adjacent regions that have a separator whose weight are smaller than a predefined threshold are merged.
4. Once all of the regions have been identified, they are organized into a tree that represents containment relationships, i.e., a region is the child of another region as long as the rendering box of the former is contained within the rendering box of the latter. The whole tree is returned by VIPS.

3.9 ViDRE: Vision-Based Web Data Records Extraction

Vision-based web Data Records Extraction [169], or ViDRE for short, relies on VIPS [24] and is intended to learn rules to extract data records from web documents. It builds on the hypothesis that a web document contains a unique data region, that this region is the largest one, it is reasonably centered, data records inside this data region are aligned to the left, adjacent, do not overlap, are separated homogeneously, and are similar from a visual point of view. This region extractor is a constituent part of the ViDE information extractor [104].

The ViDRE algorithm works as follows:

1. Given an input document, it first creates a tree of regions using VIPS [24].
2. It then identifies the data region in the previous tree by searching for the largest region that is reasonably centered horizontally. If more than one region satisfies this condition, ViDRE selects the one at the lowest level in the tree returned by VIPS.
3. The selected region is first analyzed to find noisy subregions, which are removed. The authors take only the top and bottom subregions into account, which are considered noisy as long as they are not aligned to the left.
4. The algorithm now clusters the remaining subregions using a similarity function that relies on features of the images, the text, and the links in the subregions being compared.
5. For each of the previous clusters, the algorithm first finds the minimum bounding box that encloses all of the subregions in that cluster and reorders them according to their relative positions.
6. Then, the cluster with the maximum number of subregions is selected, let it be C, and each of its subregions is considered an initial data record.
7. Now, for each cluster that is different from C, it finds the subregions whose bounding boxes overlap the bounding box of a data record in C; such subregions are merged into a unique subregion. In other words, this step grows the initial data records with overlapping subregions that are adjacent and returns them.
8. The algorithm now creates an extraction rule for the data region of the form (x, y, w, h, l), where x and y are the upper coordinates of the data region, w and h are the width and the height of this region, and l is the level of this region in the tree of regions constructed by VIPS. Furthermore, ViDRE creates an extraction rule that contains visual information regarding the first data record (font, size, etc.), and gap space between two consecutive data records. The first rule identifies the data region, whereas the second rule uses the visual information to identify the first data record, and the gap space to detect and separate the remaining data records in the data region.

3.10 RIPB: Recognizing Informative Page Blocks

Recognizing Informative Page Blocks [84], or RIPB for short, relies on VIPS [24] and is intended to identify the largest data region in a document. RIPB is supervised, which implies that the user must provide a few examples of data records. The algorithm then tries to find the regions that contain structures that are similar to these records.

The RIPB algorithm works as follows:

1. It first finds a matching between the DOM trees that represent the data records that the user must provide. This matching builds on the algorithm proposed by the DEPTA information extractor [178]. The result is a so-called augmented tree, which can be interpreted as a rule that allows to identify the trees in a document that have a data record that is similar to one of the sample data records provided by the user.
2. It then uses VIPS [24] to segment the input document into a collection of candidate regions.
3. It then applies a clustering algorithm to the previous candidate regions. The similarity function used is based on a tree-edit distance.
4. Next, each region in a cluster must be compared to the augmented tree and the result is a score that is based on the tree-edit distance. The total score of a cluster is the sum of the partial scores of its regions.
5. The algorithm selects the cluster with the highest score and returns it as the data region in the input document.

3.11 VSDR: Visual Segmentation-Based Data Record

Visual Segmentation-based Data Record [97], or VSDR for short, builds on VIPS [24] and is intended to extract data

records. It builds on the hypothesis that the regions with data records are nonleaf contiguous siblings; furthermore, it assumes that data records are totally contained within a unique region that is composed of at least two subregions with different kinds of contents.

The VSDR algorithm works as follows:

1. First, it uses VIPS [24] to partition the input document into a tree of visual regions.
2. The leaf regions returned by VIPS are explored in an attempt to classify them into the following categories: text Region, text link region, image region, image link region, drop down region, text box region, or action button region. Note that it is common that the leaf regions returned by VIPS contain a mixture of contents, e.g., text and links; this implies that VSDR may need to split them so that they fit into a unique category.
3. It then identifies noisy regions using the following heuristics: Nodes at the first level in the DOM tree that have zero or one child are removed; if at least 75 percent of the subregions inside a region were classified as text link or image link, then the region is removed; regions that are of type drop drown or action button are removed if they are rendered in a small box.
4. VSDR now attempts to merge some of the regions in the region tree returned by VIPS using two heuristics, namely: If all of the leaf subregions of a region are of the same type, then they are merged; similarly, if a region has a unique child then the parent region is removed and the child is raised to replace it in the region tree.
5. Now, the algorithm attempts to find the regions that contain the data records in the input document. To do so, VSDR compares every pair of sibling regions using a tree-edit distance algorithm. The algorithm returns the regions that are similar enough according to a user-defined threshold.

3.12 ViNTs: Visual Information and Tag Structure

Visual information aNd Tag structure [179], or ViNTs for short, is intended to learn rules to extract data records from web documents that are returned by search engines. It builds on the hypothesis that there is a unique data region that is large, centered, and contains many data records, which are very similar even if they come from different search engines, share a parent node, are clearly separated, and the shape of their left sides is very regular. ViNTs was used to generate wrappers for most of the search engines used in the AllInOneNews [101] metasearch engine

The ViNTs algorithm works as follows:

1. The input to ViNTs is a set of documents that must contain at least four data records and an additional document that does not contain any records at all.
2. First, input documents are rendered using a browser and each node is assigned the coordinates of their corresponding rendering boxes.
3. Rendering boxes are then used to extract content lines, which are classified as text, link, link-text, link-head, text-head, link-text-head, horizontal rule, and blank lines. ("Head" refers to whether the line starts with a number or not.)
4. The content lines that appear in the document that do not contain any records are removed from the other documents.
5. It then uses a suffix tree to identify repetitive line patterns that occur at least three times. Each pattern is considered as a record separator. These separators are used to partition the input document into several regions that are clustered according to their visual similarity and their type; i.e., two regions belong to the same cluster if their distance is less than a given threshold. The distance function builds on the types of lines they contain, the distance between their rendering boxes, and a so-called shape distance that measures how similar the left sides of the regions are. Each resulting cluster is a region.
6. The next step is to identify the first line of each data record in the previous regions. It builds on four simple heuristics, namely: It follows a horizontal rule, it follows a blank line, and there are no other blank lines, it is the only line that starts with a number, or it is the leftmost line. The first lines, thus, identified help identify records.
7. ViNTs now learn extraction rules of the form (p, S, α, β), where p denotes the DOM path to the region, S a set of record separators, and α and β denote the minimum and maximum number of records (NR) per region.
8. For every region, the algorithm calculates p as the longest common prefix out of the DOM paths of its records; S is calculated building on the differences among the DOM paths that lead to the records in the region being analyzed; α and β are calculated by means of a heuristic that is based on the presence of links and the similarity of the records.
9. The last step consists of selecting and merging the rules that are more likely to extract the right data records. To do so, ViNTs searches for the rules that extract a large data region that is reasonably centered and have many data records.

3.13 MSE: Multiple Section Extraction

Multiple Section Extraction [180], or MSE for short, is intended to learn rules to extract data records from all of the data regions in the documents returned by search engines. It builds on the hypothesis that data records in a data region appear consecutively, they are siblings, they are cohesive (which means that their content lines are dissimilar from each other, but records are similar as a whole), and content lines that are common to many documents do not have relevant information.

The MSE algorithm works as follows:

1. MSE takes a set of documents as input. It analyzes them using two algorithms called Multirecord Extractor, or MRE for short, and Dynamic-Section Extractor, or DSE for short. The former attempts to identify data records, whereas the latter attempts to identify data regions. Note that MRE implicitly identifies a number of data regions, as well, which

are the ones that enclose the data records this algorithm identifies. MRE is a version of ViNTs that relies on a different distance function; it then returns the same rules as ViNTs. DSE relies on the hypothesis that the boundaries of data regions consist of content lines that are not likely to change from document to document, or content lines that change very little; DSE returns rules of the form (l, r), where l denotes the left boundary of a region and r its right boundary.

2. If exactly the same region is returned by both MRE and DSE, then the records identified by MRE are selected as candidate data records; if a region returned by MRE overlaps a region returned by DSE, then the algorithm searches the DSE region for records that are similar to the records in the MRE region; if an MRE region does not overlap any DSE regions, then it is ignored; if a DSE region does not overlap any MRE regions, then the algorithm partitions it using MRE.
3. The previous step removes some MRE regions and partitions the DSE regions. They both need to be refined since the authors identified that there are chances that MRE returns so-called oversized data records, i.e., a number of consecutive actual records that are merged together, and splitted records, i.e., an actual record that is identified as two or more consecutive records.
4. To deal with oversized records, MSE iterates through the collection of regions identified by MRE, and reapplies MRE to them. If new records are found, then MSE assumes that it was oversized.
5. To deal with splitted data records, MSE checks every combination of consecutive records to find out if they are cohesive enough, i.e., they are similar as a whole, but internally dissimilar.
6. In the previous steps, MSE has partitioned every input document into a set of candidate records and data regions. It now clusters these regions using a distance function that builds on their DOM paths, their boundaries, and their tree-edit distances. Singleton clusters are ignored because they refer to regions that are isolated.
7. The algorithm now creates an extraction rule of the form (p, S, L, R) for each cluster, where p denotes the DOM path that leads to a region, S is a set of separators (in the sense of ViNTs), and L and R are the set of left and right boundaries, respectively. p is calculated by compacting the DOM paths of the regions in the same cluster; S is the union of the separators returned by MRE; and L and R are the union of the boundaries returned by DSE.

3.14 RST: Record Segmentation Tree

Record Segmentation Tree [19], or RST for short, is intended to extract data records from all of the data regions in a web document. It is based on the hypothesis that similar data records in a contiguous region compose a data region, that data records inside a data region are formatted using similar HTML tags, that a data record consists of a collection of subtrees, and that these subtrees share a parent node.

The RST algorithm works as follows:

1. The DOM tree of the input document is built, and a sequence S with its subtrees is created.
2. The subtrees sequence S is used to create a so-called RST, which is a search structure. An RST is a tree in which each node covers a subsequence of adjacent subtrees in S and has a set of separators for these subtrees. Each child node covers a subsequence of subtrees that starts at the parent subtrees and covers a number of adjacent subtrees; the separator set is the union between the parent's separators and its own set of separators.
3. Each node in the RST tree is a possible segmentation of the subtrees in S starting from a given subtree, where each segment (group of subtrees) is a data record candidate. Since data records are supposed to be formatted using similar tags, then the algorithm searches for a node in the RST such that the similarity between its segments is greater than a predefined threshold; if none is found, then the subtree considered is not a data region; if more than one node satisfies the previous similarity condition, then the node with the greatest number of data records is considered as a data region.
4. The previous step is performed starting from the first subtree in S, but if the data region is not found, the step is repeated by starting from the second subtree in S, and so on. When a data region is found, the previous step can also be applied again to the remaining subtrees to detect additional data regions in the input web document.
5. To reduce the complexity of the algorithm, building the RST is optimized by building a slimmed RST, in which some children are not created if the average number of subtrees per data record in a node is less than the number of subtrees in the child to create. Furthermore, some search pruning strategies are introduced, and noise reduction is performed by removing the first subtree in the first data record if this record contains more subtrees than the next data records and if this subtree is not similar enough to the next data records. The similarity is calculated using a token-based tree-edit distance [69], [143].

4 Comparative Analysis

We have studied and compared four dimensions of features, namely: Input and output, algorithmic, efficiency and effectiveness, and other miscellaneous features. We report on the results in the following sections.

4.1 Input and Output Dimension

This dimension contains features that are related to the input and output of region extractors, namely:

- *Domain*: It refers to the domain of the input documents. Some proposals rely on some characteristics of a specific domain to identify regions, whereas other proposals are domain independent.
- *Input*: It refers to the data on which a proposal works.
- *Number of Documents (ND)*: It refers to the minimum number of input documents that are necessary for the region extractor to work.

TABLE 4
Comparison of Input and Output Features

Proposal	Domain	Input	ND	NR	Nesting	Output	Learns rules	Rules type
Embley and others	Agnostic	HTML document	1	2+	One-level	Data records in the largest data region.	×	–
OMINI	Agnostic	HTML document	1	2+	One-level	Data records in the largest data region.	✓	Regular expression of tags.
MDR	Agnostic	HTML document	1	2+	One-level	Data records in all of the data regions.	×	–
TPC	Agnostic	HTML document	1+	2+	Multi-level	Data records in all of the data regions.	×	–
DSE	Agnostic	HTML document	1*	1+	Zero-level	All of the data regions.	×	–
U-REST	Agnostic	HTML document	1	2+	One-level	Data records in the largest data region.	×	–
STAVIES	Agnostic	HTML document	1	2+	One-Level	Data records in the largest data region.	×	–
VIPS	Agnostic	HTML document	1	1+	Multi-level	All of the regions.	×	–
ViDRE	Agnostic	Region tree	1	2+	One-level	Rules to extract data records in the largest data region.	✓	Visual delimiters.
RIPB	Agnostic	Region tree	1	1+	Zero-level	The largest data region.	×	–
VSDR	Agnostic	Region tree	1	2+	Zero-level	Data records in all of the data regions.	×	–
ViNTs	Search engines	HTML document	2+	4+	One-level	Rules to extract data records in the largest data region.	✓	Regular expression of tags.
MSE	Search engines	HTML document	2+	1+	One-level	Rules to extract data records in all of the data regions.	✓	Regular expression of tags and left and right delimiters.
RST	Agnostic	HTML document	1+	2+	Multi-level	Data records in all of the data regions.	×	–

ND = Number of documents that must be provided by the user.
NR = Number of records in the input documents.
* The user needs to provide only a document; another document is fetched automatically.

- *NR*: It refers to the minimum NR in each document that are necessary for the region extractor to work.
- *Nesting*: It refers to the ability to deal with regions that are nested, and to maintain this relationship.
- *Output*: It refers to what the region extractor returns as output.
- *Rules type*: It refers to the kind of rules produced by the region extractor, if any.

Table 4 reports on the results of our analysis regarding the previous features. Note that the majority of region extractors are domain independent, with the exception of ViNTs [179] and MSE [180], which are targeted toward search engines. The majority of the proposals need input documents to be formatted using the HTML markup language because they rely on DOM trees or HTML tags, except ViDRE [169], RIPB [84], and VSDR [97], which work on the region tree produced by VIPS [24]; this precludes them from being used with free-text documents. Most region extractors need a unique document to work. In the case of DSE [163], the user needs to provide only a document; the other is fetched automatically. ViNTs [179] and MSE [180] require several input documents, including one document that does not contain any records. Unfortunately, there is not a rule of thumb to determine what the most appropriate ND or records is; in general, the user must make an attempt to provide an example of every possible formatting so that it can be analyzed. Proposals that need more than one data record in each input document generally build on the fact that they are formatted regularly, e.g., Embley et al. [53], OMINI [23], STAVIES [125], and MSE [180], or they search for regions that contain records that are similar to the ones provided by the user, namely, RIPB [84]. Regarding nested regions, the majority of proposals are single level because they can maintain the relationship between the detected data regions and the data records inside them. VIPS [24] supports multilevel nesting because it partitions regions in a hierarchical structure in which it maintains the relationships between the parent and child subregions, thus allowing to detect nested subregions. TPC [114] and RST [19] support multilevel nesting because they are able to detect nested regions and to infer the relationship between parent and child regions. Note that proposals that use VIPS, such as RIPB [84] and VSDR [97], are not considered to extract nested data regions (zero-level) because the former extracts one data region only and the latter does not maintain the relationships between regions. DSE [163] is not considered to extract nested data regions since it does not extract the data records from the detected data regions. Once the largest data region or all of the data regions are detected, some proposals partition them into records that are then output, e.g., Embley et al. [53], MDR [100], and STAVIES [125]. Some region extractors produce extraction rules, namely: OMINI [23], [169], ViNTs [179] and MSE [180]. These rules range from regular

TABLE 5
Comparison of Algorithmic Features

Proposal	Supervised	View	Features	Algorithm	Removes noise regions
Embley and others	×	DOM tree	Tag information: Node with highest number of children, common separating tags, tag frequencies.	Stanford's certainty theory to combine heuristics.	×
OMINI	×	DOM tree	Tag information: Node with highest number of children, common separating tags, tag frequencies, sibling nodes, and DOM paths	Probabilistic method to combine heuristics.	✓
MDR	×	DOM tree	Tag information: Repetitive similar structures.	Combinatorial algorithm; String matching	×
TPC	×	DOM tree	Tag information: DOM path similarity. Visual information: Height and width of some nodes.	Spectral clustering	×
DSE	×	DOM tree	Tag information: Similarity between DOM nodes.	Tree matching	×
U-REST	~	DOM tree	Tag information: Similarity between DOM nodes.	Clustering; Tree matching; String matching	✓
STAVIES	×	DOM tree	Tag information: Similarity between leaf nodes.	Hierarchical clustering	✓
VIPS	×	Rendering boxes DOM tree	Visual information: Position, background colour, foreground colours, font information, background image, and nodes rendering boxes.	Hierarchical clustering	✓
ViDRE	×	Regions returned by VIPS	Visual information: Position, layout, appearance, contents.	Clustering	✓
RIPB	✓	Regions returned by VIPS	Tag information: Similarity between DOM nodes. Visual information: The same used by VIPS.	Clustering; Tree matching	✓
VSDR	×	Regions returned by VIPS	Tag information: Similarity between DOM nodes. Visual information: The same used by VIPS.	Tree matching	✓
ViNTs	×	Content lines	Tag information: DOM trees. Visual information: rendering boxes	String matching; Suffix trees; Shape distance; Position distance	×
MSE	×	Content lines	Tag information: DOM trees. Visual information: rendering boxes.	Clustering; Tree matching	✓
RST	×	DOM tree	Tag information: Similar structures.	Token-based edit distance	×

expressions to tuples that contain visual or tag information to locate a data region and separate it into data records.

4.2 Algorithmic Dimension

The second dimension includes features that are related to the algorithms used in the region extractors. These features are the following:

- *Supervised*: It refers to whether a proposal requires the user to provide samples of the regions in which he or she is interested.
- *View*: It refers to the view of the input document on which a region extractor works.
- *Features*: It refers to the features of the web document used by the region extractor.
- *Algorithm*: It refers to the algorithms used by the region extractor.
- *Removes noise*: It refers to whether a proposal implements an algorithm to remove noisy regions or not.

Table 5 reports on the results of our analysis regarding algorithmic features. The majority of proposals are unsupervised, except for RIPB [84], which requires the user to provide a few examples of the records in which he or she is interested. Note that U-REST [141] is considered an unsupervised proposal, but it requires a similarity and a ranking function to be learned from examples; the key is that these functions need not to be customized for specific sites, i.e., they can be learned once and reused many times. The proposal by Embley et al. [53] needs an ontology to match data records, but the algorithm still works unsupervisedly if this ontology is not provided, although this may have an impact on its effectiveness. Approximately half of the proposals work on the DOM tree of the input documents; a quarter of the proposals rely on the regions

TABLE 6
Standard Measures to Characterize the Effectiveness of a Classifier

	Expected regions		
Extracted regions	tp = true positive	fp = false positive	$P = \frac{tp}{tp+fp}$
	fn = false negative	tn = true negative	$N = \frac{tn}{fn+tn}$
	$R = \frac{tp}{tp+fn}$	$S = \frac{tn}{fp+tn}$	$A = \frac{tp+tn}{tp+fp+fn+tn}$

P = Precision
R = Recall
S = Specificity
N = Negative Predictive Value
A = Accuracy

that were returned by VIPS, whereas ViNTs [179] and MSE [180] convert input documents into collections of so-called content lines. Region extractors are based on two types of features, namely: Visual and tag information. Note that half of the proposals use only tag information, e.g., OMINI [23] and RST [19]; ViDRE [169] is the unique region extractor that is solely based on visual information, whereas the remaining proposals use both visual and tag information, e.g., ViNTs [179] and TPC [114]. Note that the majority of proposals rely on a number of key algorithms in the literature: Tree matching, string matching, and clustering using a variety of similarity functions. The majority of proposals take noise reduction into account by filtering some regions that are not likely to be a data region, which is performed by removing small regions, regions that do not contain similar-enough structures, regions that appear in almost every document, or regions that are isolated.

4.3 Efficiency and Effectiveness Dimension

The third dimension includes efficiency and effectiveness features. Note that the problem of identifying regions is somewhat related to the problem of setting up a binary classifier that works on all of the possible regions in a document and classifies them as either relevant or not. It is not surprising then that the usual measures to characterize the effectiveness of a region extractor are the standard measures used to characterize the effectiveness of a binary classifier. Table 6 provides the exact formulation of these measures:

- *Complexity/Time*: Complexity refers to the computational complexity of a proposal; as usual, this is characterized using the well-known big-O notation. Since the majority of proposals do not report their complexity, we added the Time feature that reports on the learning and extraction times if they are reported in the original proposal.
- *Test data sets*: It refers to the data sets that were used to test a proposal experimentally, and the features of the web documents inside the ad hoc data sets.
- *Compared with*: It refers to the techniques with which each proposal was compared.
- *Precision* (P): It refers to the average fraction of regions that are identified by a proposal and correspond to actual data regions. Intuitively, the higher the precision, the less regions identified by a proposal are not actually relevant.
- *Recall* (R): It refers to the average fraction of actual data regions that are identified by a proposal; intuitively, the higher the recall, the more actual data regions are identified.
- *Accuracy* (A): It focuses on both the ability of a proposal to identify data regions and not to identify ancillary regions.

Table 7 reports on the results of our analysis regarding efficiency and effectiveness. Note that this is by far the dimension in which there is more missing data in the literature. Unfortunately, only a few proposals are accompanied by a complexity analysis, namely: Embley et al. [53], MDR [100], DSE [163], TPC [114], and [19]. Some proposals reported on the learning and extraction times, namely: STAVIES [125], ViDRE [169], ViNTs [179], and MSE [180]. However, the timings reported in these proposals are not side-by-side comparable because they were measured on different machines. The majority of proposals have been analyzed on ad hoc data sets, which reflects the lack of standardized up-to-date data sets; some proposals used the TBDW [173] repository and the data sets used in MEMPHIS ([112]) project, which are not maintained since 2004. Comparisons are fairer in the cases in which third-party data sets were used, namely: TPC Miao et al. [114], STAVIES [125], VSDR [97] and MSE [180]. Regarding the standard measures P, R, and A, note that the results reported in Table 7 are not comparable side-by-side because they were calculated on different data sets; this is an important problem in this field, because this makes it impossible to discern which proposal globally behaves better than the others. Furthermore, note that the proposals surveyed generally achieve very good results when test data sets are ad hoc, namely: Embley et al. [53], OMINI [23], MDR [100], DSE [163], U-REST [141], ViDRE [169], RIPB [84], ViNTs [179], MSE [180], and RST [19], but these measures are lower in case the data sets are third-party data sets, namely: TPC [114], STAVIES [125], and VSDR [97]. Note that although MSE [180] uses the ViNTs [179] data sets, which are not considered third-party data sets because both proposals were presented by the same authors. None of the proposals we have surveyed reported on specificity or on the negative prediction, which are other well-known effectiveness measures in the field of machine learning.

4.4 Miscellaneous Dimension

The fourth dimension includes the following features:

- *Availability*: It refers to whether an implementation is publicly available to the research community or not.
- *Used in/Inspired*: It refers to the techniques in the literature that use a proposal or whose region extractor was inspired by a proposal.

TABLE 7
Comparison of Effectiveness and Efficiency Features

Proposal	Complexity/Time	Test datasets	Compared with	P	R	A
Embley and others	$O(n)$, where n it the total number of nodes in the web document.	Web documents that contain one data region composed of multiple data records.	***	–	–	100.00%
OMINI	–	Web documents that contain one data region composed of multiple data records.	Embley and others [52]	100.00%	94.00%	–
MDR	$O(n\,k^2)$, where n is the total number of nodes and k is number of combinations.	Web documents that contain data regions formatted using HTML tables.	OMINI [22]; IEPAD [34]	100.00%	99.80%	–
TPC	$O(l\,m) + O(m^3)$, where l is the total number of tags and m is the number of unique DOM paths in the input web document.	Web documents that contain data regions in which there are flat or nested data records. The technique was also tested on the TBDW [172] repository.	MDR [98]	96.23%	97.03%	–
DSE	$O(h\,m^2)$, where h is the height of the DOM tree and m is the maximum number of nodes in a level.	Web documents that contain one data region composed of multiple data records.	—	–	–	100.00%
U-REST	–	Web documents that contain one data region composed of multiple data records.	OMINI [22]; MDR [98]	–	92.73%*	98.46%*
STAVIES	It takes less than 0.1 seconds when the web documents are simple and less than 0.4 seconds when the documents are complex on a Pentium-4 1.7GHz PC with 512 MB RAM.	Web documents that contain one data region composed of multiple data records. The technique was also tested on the datasets used in MEMPHIS [112] project and the datasets used to test MDR [98] and OMINI [22]	MDR [98]; OMINI [22]	90.58%	94.12%	–
VIPS	–	The technique was tested on the WT10G [35] and .GOV [154] repositories.	***	–	–	–
ViDRE	It takes no more than 2 seconds to extract data records from web documents with no more than 20 data records on a Pentium-4, 2GHz PC.	Web documents that contain one data region composed of at least two data records.	MDR [98]	98.70%	97.20%	–
RIPB	–	Web documents that contain one or more data records from shopping and news web sites.	–	98.13%**	95.78%**	–
VSDR	–	It was tested on the TBDW [172] repository and the same datasets used to test ViNTs [178].	ViNTs [178],MDR [98]	89.94%	97.60%	–
ViNTs	It takes between 3 and 7 seconds to learn extraction rules from a 6-document dataset and 0.1 to apply them on a Pentium-4 1.7GHz PC.	Web documents returned by search engines, that contain a unique data region, which is composed of multiple data records.	MDR [98]	98.90%	98.30%	–
MSE	It takes between 20 and 50 seconds to learn extraction rules from 5 web documents on a Pentium-M 1.3GHz laptop.	Web documents returned by search engines, that contain multiple data regions. The technique was also tested on the datasets used to test ViNTs [178].	***	98.80%	98.70%	–
RST	$O(s^2)$, where s is the number of subtrees in an input web document.	Web documents that contain data regions in which resides nested data records. The technique was also tested on the TBDW [172] repository and the datasets used to test ViNTs [178].	MDR [98], ViNTs [178], TPC [114]	98,06%	97,88%	–

* We report on the best results; the original paper reports on several results using different techniques to calculate the similarity between clusters.
** We report on approximate results since the original paper provides them in a bar chart, which makes it difficult to know the exact figures.
*** The authors compare different methods that were devised by themselves and a combined approach between these methods, which is their proposal.

- *Applicability*: It characterizes the situation in which a proposal is most applicable.
- *Weak features*: It reports on known limitations and weaknesses of a proposal.

Table 8 reports on the results of our analysis regarding miscellaneous features. Unfortunately, only a few proposals are available on the web to the research community. Half of them are used or have inspired other proposals. Note that about half of the proposals are applicable to semistructured web documents that contain multiple data regions, whereas the other half needs web documents to contain a unique data region. The main drawbacks of the studied proposals

TABLE 8
Comparison of Miscellaneous Features

Proposal	Availability	Used in or inspired by	Applicability	Weak features
Embley and others	×	OMINI [22]	Semi-structured web documents that contain a unique data region that contains several flat and similar data records.	It fails in cases in which a document contains several data regions, large menus or long listings of user comments, or a unique data record. Buttler and others [22] identified that the technique does not work well if the user-defined ontology on which it relies is omitted.
OMINI	✓[1]	XWRAP Elite [70]	Semi-structured web documents that contain a unique data region that contains several flat and similar data records.	It fails in cases in which a document contains several data regions, large menus or long listings of user comments, or a unique data record.
MDR	✓[2]	DEPTA [177], NET [99], ViPER [143]	Semi-structured web documents that contain one or more data regions formatted using HTML table tags, such that each data region contains several similar flat data records.	MDR fails in documents that have large menus, long listings of user comments, or documents in which the relevant information is rendered using lists, divisors, or other HTML constructs.
TPC	×	–	Semi-structured web documents that contain one or more data regions, such that each data region contains several flat or nested similar data records.	TPC fails in cases in which a data region contains a unique data record, data records are composed of less than tree tags, or in documents with large menus in which each item is composed of more than three tags.
DSE	×	DeLa [164]	Semi-structured web documents that come from a web site that consists of more than one section, so that ancillary regions in documents from different sections are similar and data regions are not.	DSE fails in cases in which a web site consists of a unique section within which the documents are very similar from a structural point of view.
U-REST	×	–	Semi-structured web documents that contain a unique data region that contains several flat and similar data records.	U-REST fails in cases in which a web document contains more than one information region, contains a unique record, or in cases in which the same document shows information about several products with different formattings.
STAVIES	×	–	Semi-structured web documents that contain a unique data region that must contain at least 20% of the data of the document and several flat and similar data records.	STAVIES fails in cases in which a document has a menu that is bigger than the data region, cases in which there are several data regions, and cases in which a document contains a unique record.
VIPS	✓[3]	ViDRE [102]; RIPB [83]; VSDR [96]; MTPE [97]	Web documents in which the visual features of a region allow to separate it from other regions.	VIPS cannot discern which the data region is, which implies that this technique returns headers, footers, banners, menus and other regions that are not usually relevant.
ViDRE	×	ViDE [103]	Web documents in which the data region is the largest one, centred, and data records inside the data region are flat, aligned to the left, adjacent, do not overlap, are separated homogeneously, and are similar from a visual point of view.	ViDRE fails in cases in which a document contains several data regions that are separated by banners, for instance, or cases in which the text flows from right to left.
RIPB	×	–	Web documents whose data regions contain flat data records that are similar to the data records provided by user.	RIPB fails in cases in which a data record spans several regions and in cases in which a document has several data regions.
VSDR	×	–	Web documents whose data records are flat, similar and are composed of several types of information, eg., links, images and text.	VSDR fails in cases in which a data record spans several regions or cases in which records are very simple since they consist of only text, images, or lists of links, to mention a few examples.
ViNTs	✓[4]	MSE [179]	Web documents returned by search engines that contain one data region in which there are three or more similar and flat data records.	ViNTs fails when input documents are written in languages that are not typeset from left to right and it may return ancillary regions if they are large enough.
MSE	×	–	Web documents returned by search engines that contain one or more data regions in which there are one or more similar and flat data records.	MSE fails with search engines that return records about several topics; for instance, Google may return records about web documents, about images, news, related topics and searches, which makes them very dissimilar from each other. It also fails when records in a data region are not siblings.
RST	×	–	Semi-structured web documents that contain one or more data regions, such that each data region contains several similar flat or nested data records.	RST fails in cases in which a data region contains a unique data record, or the records in a data region are presented using different formatting tags.

[1] http://www.cc.gatech.edu/projects/disl/Omini/index.html
[2] http://www.cs.uic.edu/~liub/WebDataExtraction/MDR-download.html
[3] http://www.zjucadcg.cn/dengcai/VIPS/VIPS.html
[4] http://www.data.binghamton.edu/vints/demo.html

are that many of them build on the hypothesis that the input web document contains a unique data region (Embley et al., OMINI, TPC, U-REST, STAVIES, ViDRE, and RIPB); furthermore, many proposals are based on the hypothesis that a data region contains multiple similar data records (Embley et al., OMINI, MDR, TPC, U-REST, STAVIES, ViDRE, VSDR, ViNTs, MSE, and RST), which fails when a web document contains a unique data record with details about a unique product or service or multiple data records with different formatting tags; furthermore, many proposals are based on hypothesis regarding the size of the data region (Embley et al., OMINI, TPC, STAVIES, and ViNTs),

which fails when data regions are smaller than ancillary regions. Only few proposals can be applied to extract nested data records and their relationships. Note, too, that none of the proposals can be considered universally applicable due to their weak points.

5 Conclusions

Web documents are getting more and more sophisticated, which complicates the task of information extraction. This has motivated using region extractors so that information extractors can focus only on data records or data regions. The importance of data regions extractors is clear given that some of the proposals we have analyzed are now an intrinsic part of recent information extractors or have inspired them; there are also applications to fields such as information retrieval, focused web crawling, topic distillation, adaptive content delivery, mashups, and metasearch engines.

Despite the importance of the region extractors, they have not been compared extensively in the literature. In this paper, we have surveyed region extractors and compared them regarding four dimensions, namely: Input and output, algorithmic, efficiency and effectiveness, and some miscellaneous features. The following conclusion can be drawn from our survey:

1. All of the region extractors work on semistructured documents that are formatted in HTML and rely on their DOM tree directly or indirectly by using VIPS. In general, they search for repetitive structures to identify data regions. This makes it difficult to apply them to free-text documents whose contents do not rely heavily on HTML tags.
2. The majority of region extractors are unsupervised and usually rely on the following algorithms: Tree matching, string matching, and clustering. This makes the majority of proposals scalable because they do not rely on a user to provide samples of the regions to be extracted.
3. The efficiency and effectiveness is by far the dimension in which there are more missing data in the literature; furthermore, the available results are not comparable side by side. It is an essential requirement to count on an up-to-date and maintained data set repository to perform homogeneous and fair empirical evaluations.
4. Region extraction is not an easy task. The proposals in the literature have strong features and drawbacks, but none of them is universally applicable, which keeps this quite an active research field.

Acknowledgments

This work was supported by the European Commission (FEDER), the Spanish and the Andalusian R& D& I programmes (Grants TIN2007-64119, P07-TIC-2602, P08-TIC-4100, TIN2008-04718-E, TIN2010-21744, TIN2010-09809-E, TIN2010-10811-E, and TIN2010-09988-E).

References

[1] B. Adelberg, "NoDoSE: A Tool for Semi-Automatically Extracting Semi-Structured Data from Text Documents," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 283-294, 1998.

[2] E. Agichtein and V. Ganti, "Mining Reference Tables for Automatic Text Segmentation," *Proc. ACM 10th SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 20-29, 2004.

[3] E. Agichtein and L. Gravano, "Snowball: Extracting Relations from Large Plain-Text Collections," *Proc. Int'l Conf. Digital Libraries (ICDL),* pp. 85-94, 2000.

[4] J.S. Aitken, "Learning Information Extraction Rules: An Inductive Logic Programming Approach," *Proc. 15th European Conf. Artificial Intelligence (ECAI),* pp. 355-359, 2002.

[5] E. Almasy, D. Sleasman, and R. Bower, "Software for Building a Full-Featured Discipline-Based Web Portal: The Scout Portal Toolkit," *D-Lib Magazine,* vol. 8, no. 11, p. 11, 2002, http://dx.doi.org/10.1045/november2002-almasy.

[6] M. Álvarez, A. Pan, J. Raposo, F. Bellas, and F. Cacheda, "Extracting Lists of Data Records from Semi-Structured Web Pages," *Data Knowledge Eng.,* vol. 64, no. 2, pp. 491-509, 2008.

[7] D.E. Appelt, J.R. Hobbs, J. Bear, D.J. Israel, and M. Tyson, "FASTUS: A Finite-State Processor for Information Extraction from Real-World Text," *Proc. 13th Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 1172-1178, 1993.

[8] A. Arasu and H. Garcia-Molina, "Extracting Structured Data from Web Pages," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 337-348, 2003.

[9] J.L. Arjona, R. Corchuelo, D. Ruiz, and M. Toro, "From Wrapping to Knowledge," *IEEE Trans. Knowledge Data Eng.,* vol. 19, no. 2, pp. 310-323, Feb. 2007.

[10] G.O. Arocena and A.O. Mendelzon, "WebOQL: Restructuring Documents, Databases, and Webs," *TAPOS,* vol. 5, no. 3, pp. 127-141, 1999.

[11] J.H. Aseltine, "WAVE: An Incremental Algorithm for Information Extraction," *Proc. AAAI Workshop Machine Learning for Information Extraction,* 1999.

[12] N. Ashish and C.A. Knoblock, "Semi-Automatic Wrapper Generation for Internet Information Sources," *Proc. Conf. Cooperative Information Systems (CoopIS),* pp. 160-169, 1997.

[13] F. Ashraf, T. Özyer, and R. Alhajj, "Employing Clustering Techniques for Automatic Information Extraction from HTML Documents," *IEEE Trans. Systems, Man, and Cybernetics, Part C,* vol. 38, no. 5, pp. 660-673, Sept. 2008.

[14] P. Atzeni and G. Mecca, "Cut & Paste," *Proc. ACM 16th SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems (PODS),* pp. 144-153, 1997.

[15] R. Basili, M.T. Pazienza, M. Vindigni, P. Bank and The Reuters Trevi Collection, "Corpus-Driven Learning of Event Recognition Rules," *Proc. ECAI Workshop Machine Learning for Information Extraction,* 2000.

[16] R. Baumgartner, S. Flesca, and G. Gottlob, "Visual Web Information Extraction with Lixto," *Proc. 27th Int'l Conf. Very Large Data Bases (VLDB),* pp. 119-128, 2001.

[17] G. Beuster, B. Thomas, and C. Wolff, "MIA: An Ubiquitous Multi-Agent Web Information System," *Proc. Int'l ICSC Symp. Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce,* pp. 11-13, 2000.

[18] D.M. Bikel, S. Miller, R.M. Schwartz, and R.M. Weischedel, "Nymble: A High-Performance Learning Name-Finder," *Proc. Fifth Conf. Applied Natural Language Processing (ANLP),* pp. 194-201, 1998.

[19] L. Bing, W. Lam, and Y. Gu, "Towards a Unified Solution: Data Record Region Detection and Segmentation," *Proc. 20th ACM Int'l Conf. Information and Knowledge Management (CIKM),* pp. 1265-1274, 2011.

[20] V.R. Borkar, K. Deshmukh, and S. Sarawagi, "Automatic Segmentation of Text into Structured Records," *Proc. ACM SIGMOD Int'l Conf. Management of Data* pp. 175-186, 2001.

[21] S. Brin, "Extracting Patterns and Relations from the World Wide Web," *Proc. Selected Papers Int'l Workshop World Wide Web and Databases (WebDB),* pp. 172-183, 1998.

[22] R.C. Bunescu, R. Ge, R.J. Kate, E.M. Marcotte, R.J. Mooney, A.K. Ramani, and Y.W. Wong., "Comparative Experiments on Learning Information Extractors for Proteins and their Interactions," *Artificial Intelligence in Medicine,* vol. 33, no. 2, pp. 139-155, 2005.

[23] D. Buttler, L. Liu, and C. Pu, "A Fully Automated Object Extraction System for the World Wide Web," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 361-370, 2001.

[24] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," *Proc. Fifth Asia Pacific Web Conf. (APWeb),* pp. 406-417, 2003.

[25] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Block-Based Web Search," *Proc. 27th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR),* pp. 456-463, 2004.

[26] M.E. Califf and R.J. Mooney, "Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction," *J. Machine Learning Research,* vol. 4, 177-210, 2003.

[27] A. Carlson and C. Schafer, "Bootstrapping Information Extraction from Semi-Structured Web Pages," *Proc. European Conf. Machine Learning and Knowledge Discovery in Databases (ECML/PKDD),* pp. 195-210, 2008.

[28] N. Català, N. Castell, and M. Martín, "Essence: A Portable Methodology for Acquiring Information Extraction Patterns," *Proc. 14th European Conf. Artificial Intelligence (ECAI),* pp. 411-415, 2000.

[29] J.Y. Chai and A.W. Biermann, "The Use of Lexical Semantics in Information Extraction," *Proc. ACL Workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications,* pp. 61-70, 1997.

[30] J.Y. Chai, A.W. Biermann, and C.I. Guinn, "Two Dimensional Generalization in Information Extraction," *Proc. 16th Nat'l Conf. Artificial Intelligence and 11th Innovative Applications Artificial Intelligence Conf. Innovative Applications of Artificial Intelligence (AAAI/IAAI),* pp. 431-438, 1999.

[31] S. Chakrabarti, "Integrating the Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information Extraction," *Proc. 10th Int'l Conf. World Wide Web (WWW),* pp. 211-220, 2001.

[32] S. Chakrabarti, K. Punera, and M. Subramanyam, "Accelerated Focused Crawling through Online Relevance Feedback," *Proc. 11th Int'l Conf. World Wide Web (WWW),* pp. 148-159, 2002.

[33] C.-H. Chang and S.-C. Kuo, "OLERA: Semisupervised Web-Data Extraction with Visual Support," *IEEE Intelligent Systems,* vol. 19, no. 6, pp. 56-64, Nov./Dec. 2004.

[34] C.-H. Chang and S.-C. Lui, "IEPAD: Information Extraction Based on Pattern Discovery," *Proc. 10th Int'l Conf. World Wide Web (WWW),* pp. 681-688, 2001.

[35] C.-H. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan, "A Survey of Web Information Extraction Systems," *IEEE Trans. Knowledge Data Eng.,* vol. 18, no. 10, pp. 1411-1428, Oct. 2006.

[36] W.-T. Milly Chiang, M. Hagenbuchner, and A.C. Tsoi, "The WT10G Data Set and the Evolution of the Web," *Proc. 14th Int'l Conf. World Wide Web (WWW) (Special Interest Tracks and Posters),* pp. 938-939, 2005.

[37] H.L. Chieu and H.T. Ng, "A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text," *Proc. 18th Nat'l Conf. Artificial Intelligence (AAAI),* pp. 786-791, 2002.

[38] H.L. Chieu, H.T. Ng, and Y.K. Lee, "Closing the Gap: Learning-Based Information Extraction Rivaling Knowledge-Engineering Methods," *Proc. 41st Ann. Meet. Associ. Computational Linguistics (ACL),* pp. 216-223, 2003.

[39] Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan, "Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns," *Pro. Conf. Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP),* 2005.

[40] F. Ciravegna, "Adaptive Information Extraction from Text by Rule Induction and Generalisation," *Proc. 17th Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 1251-1256, 2001.

[41] W.W. Cohen, M. Hurst, and L.S. Jensen, "A Flexible Learning System for Wrapping Tables and Lists in HTML Documents," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 232-241, 2002.

[42] Content Extractor, "Content Extractor (Home Page)," http://integration.pervasive.com, Jan. 2012.

[43] C. Cox, J. Nicolson, J.R. Finkel, C. Manning, and P. Langley, "Template Sampling for Leveraging Domain Knowledge in Information Extraction," *Proc. PASCAL Challenges Workshop,* 2005.

[44] V. Crescenzi and G. Mecca, "Grammars Have Exceptions," *Information Systems,* vol. 23, no. 8, pp. 539-565, 1998.

[45] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 109-118, 2001.

[46] C. Cumby and D. Roth, "Feature Extraction Languages for Propositionalzed Relational Learning," *Proc. IJCAI Workshop Learning Statistical Models from Relational Data,* pp. 24-31, 2003.

[47] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications," *Proc. Meeting Assoc. for Computational Linguistics,* pp. 1-8, 2002.

[48] N.N. Dalvi, A. Machanavajjhala, and B. Pang, "An Analysis of Structured Data on the Web," *Proc. VLDB Endowment,* vol. 5, no. 7, pp. 680-691, 2012.

[49] Denodo, "Denodo (Home Page)," http://www.denodo. com, Jan. 2012.

[50] R.B. Doorenbos, O. Etzioni, and D.S. Weld, "A Scalable Comparison-Shopping Agent for the World Wide Web," *Proc. First Int'l Conf. Autonomous Agents,* pp. 39-48, 1997.

[51] H. Elmeleegy, J. Madhavan, and A.Y. Halevy, "Harvesting Relational Tables from Lists on the Web," *Proc. VLDB Endowment,* vol. 2, no. 1, pp. 1078-1089, 2009.

[52] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, Y.-K. Ng, D. Quass, and R.D. Smith, "Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages." *Data Knowledge Eng.,* vol. 31, no. 3 pp. 227-251, 1999.

[53] D.W. Embley, Y.S. Jiang, and Y.-K. Ng, "Record-Boundary Discovery in Web Documents," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 467-478, 1999.

[54] R. Feldman, B. Rosenfeld, and M. Fresko, "TEG: A Hybrid Approach to Information Extraction," *Knowledge Information Systems,* vol. 9, no. 1, pp. 1-18, 2006.

[55] D. Ferrucci and A. Lally, "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment," *Natural Language Eng.,* vol. 10, no. 3/4, pp. 327-348, 2004.

[56] Fetch Technologies, "Fetch Technologies (Home Page)," http://www.fetch.com, Jan. 2012.

[57] A. Finn and N. Kushmerick, "Information Extraction by Convergent Boundary Classification," *Proc. AAAI Workshop Adaptive Text Extraction and Mining,* 2004.

[58] First Rain, "First Rain (Home Page)," http://www.firstrain. com, Jan. 2012.

[59] D. Freitag, "Information Extraction from HTML: Application of a General Machine Learning Approach," *Proc. 15th Nat'l Conf. Artificial Intelligence/10th Conf. Innovative Applications of Artificial Intelligence (AAAI/IAAI),* pp. 517-523, 1998.

[60] D. Freitag, "Toward General-Purpose Learning for Information Extraction," *Proc. 17th Int'l Conf. Computational Linguistics (COLING-ACL),* pp. 404-408, 1998.

[61] D. Freitag and N. Kushmerick, "Boosted Wrapper Induction," *Proc. 17th Nat'l Conf. Artificial Intelligence/12th Conf. Innovative Applications of Artificial Intelligence (AAAI/IAAI),* pp. 577-583, 2000.

[62] D. Freitag and A. McCallum, "Information Extraction with HMM Structures Learned by Stochastic Optimization," *Proc. 17th Nat'l Conf. Artificial Intelligence and 12th Conf. Innovative Applications of Artificial Intelligence (AAAI),* pp. 584-589, 2000.

[63] D. Freitag and A.K. Mccallum, "Information Extraction with HMMs and Shrinkage," *Proc. AAAI Workshop Machine Learning for Information Extraction,* pp. 31-36, 1999.

[64] D.G. Gregg and S. Walczak, "Exploiting the Information Web," *IEEE Trans. Systems, Man, and Cybernetics, Part C,* vol. 37, no. 1, pp. 109-125, Jan. 2007.

[65] G. Grieser, K.P. Jantke, S. Lange, and B. Thomas, "A Unifying Approach to HTML Wrapper Representation and Learning," *Proc. Third Int'l Conf. Discovery Science,* pp. 50-64, 2000.

[66] P. Gulhane, R. Rastogi, S.H. Sengamedu, and A. Tengli, "Exploiting Content Redundancy for Web Information Extraction," *Proc. 19th Int'l Conf. World Wide Web (WWW),* pp. 1105-1106, 2010.

[67] P. Gulhane, A. Madaan, R.R. Mehta, J. Ramamirtham, R. Rastogi, S. Satpal, S.H. Sengamedu, A. Tengli, and C. Tiwari, "Web-Scale Information Extraction with Vertex," *Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE),* pp. 1209-1220, 2011.

[68] R. Gupta and S. Sarawagi, "Answering Table Augmentation Queries from Unstructured Lists on the Web," *Proc. VLDB Endowment,* vol. 2, no. 1, pp. 289-300, 2009.

[69] D. Gusfield and J. Stoye, "Linear Time Algorithms for Finding and Representing all the Tandem Repeats in a String," *J. Computer Systems Sciences,* vol. 69, no. 4, pp. 525-546, 2004.

[70] J. Hammer, J. McHugh, and H. G.-Molina, "Semistructured Data: The Tsimmis Experience," *Proc. First East-European Workshop Advances in Databases and Information Systems (ADBIS),* pp. 1-8, 1997.

[71] W. Han, D. Buttler, and C. Pu, "Wrapping Web Data into XML," *SIGMOD Record,* vol. 30, no. 3, pp. 33-38, 2001.

[72] J. Higgins and S. Keller-McNulty, *Concepts in Probability and Stochastic Modeling.* Duxbury Press, 1995.

[73] A.W. Hogue and D.R. Karger, "Thresher: Automating the Unwrapping of Semantic Content from the World Wide Web," *Proc. 14th Int'l Conf. World Wide Web (WWW),* pp. 86-95, 2005.

[74] J.L. Hong, E.-G. Siew, and S. Egerton, "Information Extraction for Search Engines Using Fast Heuristic Techniques," *Data Knowledge Eng.,* vol. 69, no. 2, pp. 169-196, 2010.

[75] C.-N. Hsu and M.-T. Dung, "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web," *Information Systems,* vol. 23, no. 8, pp. 521-538, 1998.

[76] G. Huck, P. Fankhauser, K. Aberer, and E.J. Neuhold, "Jedi: Extracting and Synthesizing Information from the Web," *Proc. Conf. Cooperative Information Systems (CoopIS),* pp. 32-43, 1998.

[77] S.B. Huffman, "Learning Information Extraction Patterns from Examples," *Proc. Connectionist, Statistical, and Symbolic Approaches Learning for Natural Language Processing,* pp. 246-260, 1995.

[78] Intelligent Miner, "IBM Intelligent Miner (Home Page)," http://www.ibm.com/developerworks/data/library/tutorials/iminer/iminer.html. Jan. 2012.

[79] T.S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu, "The Avatar Information Extraction System," *IEEE Data Eng. Bull.,* vol. 29, no. 1, pp. 40-48, Month 2006.

[80] J.Y.j. Hsu and W.t. Yih, "Template-Based Information Mining from HTML Documents," *Proc. 14th Nat'l Conf. Artificial Intelligence/Ninth Conf. Innovative Applications Artificial Intelligence (AAAI/IAAI),* pp. 256-262, 1997.

[81] M. Junker, M. Sintek, and M. Rink, "Learning for Text Categorization and Information Extraction with ILP," *Proc. Workshop Learning Language in Logic,* pp. 247-258, 1999.

[82] E. Kaasinen, M. Aaltonen, J. Kolari, S. Melakoski, and T. Laakko, "Two Approaches to Bringing Internet Services to WAP Devices," *Computer Networks,* vol. 33, no. 1-6, pp. 231-246, 2000.

[83] N. Kambhatla, "Combining Lexical Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations," *Proc. ACL (Interactive Poster & Demonstration Sessions),* 2004.

[84] J. Kang and J. Choi, "Recognising Informative Web Page Blocks Using Visual Segmentation for Efficient Information Extraction," *J. Universal Computer Science,* vol. 14, no. 11, pp. 1893-1910, 2008.

[85] M. Kayed and C.-H. Chang, "FiVaTech: Page-Level Web Data Extraction from Template Pages," *IEEE Trans. Knowledge Data Eng.,* vol. 22, no. 2, pp. 249-263, Feb. 2010.

[86] J.-T. Kim and D.I. Moldovan, "Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction," *IEEE Trans. Knowledge Data Eng.,* vol. 7, no. 5, pp. 713-724, Oct. 1995.

[87] T. Kistler and H. Marais, "WebL: A Programming Language for the Web," *Computer Networks,* vol. 30, no. 1-7 pp. 259-270, 1998.

[88] D. Klein and C.D. Manning, "Conditional Structure Versus Conditional Estimation in NLP Models," *Proc. ACL Conf. Empirical Methods in Natural Language Processing,* pp. 9-16, 2002.

[89] R. Kosala, H. Blockeel, M. Bruynooghe, and J.V.d. Bussche, "Information Extraction from Structured Documents Using *k*-Testable Tree Automaton Inference," *Data Knowledge Eng.,* vol. 58, no. 2, pp. 129-158, 2006.

[90] G.R. Krupka, "SRA: Description of the SRA System as Used for MUC-6," *Proc. Sixth Conf. Message Understanding (MUC),* pp. 221-235, 1995.

[91] S. Kuhlins and R. Tredwell, "Toolkits for Generating Wrappers," *Proc. Revised Papers Int'l Conf. NetObjectDays Objects, Components, Architectures, Services, and Applications Networked World,* pp. 184-198, 2002.

[92] N. Kushmerick and B. Thomas, "Adaptive Information Extraction: Core Technologies for Information Agents," *Agent Link,* pp. 79-103, 2003.

[93] N. Kushmerick, D.S. Weld, and R.B. Doorenbos, "Wrapper Induction for Information Extraction," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 729-737, 1997.

[94] A.H.F. Laender, B.A. Ribeiro-Neto, and A. Soares da Silva, "DEByE: Data Extraction by Example," *Data Knowledge Eng.,* vol. 40, no. 2, 121-154, 2002.

[95] A.H.F. Laender, B.A. Ribeiro-Neto, A. Soares da Silva, and J.S. Teixeira, "A Brief Survey of Web Data Extraction Tools," *SIGMOD Record,* vol. 31, no. 2, pp. 84-93, 2002.

[96] W.G. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C. Dolan, and S. Goldman, "Description of the CIRCUS System used for MUC-5," *Proc. Conf. Message Understanding (MUC),* pp. 277-291, 1993.

[97] L. Li, Y. Liu, A. Obregon, and M. Weatherston, "Visual Segmentation-Based Data Record Extraction from Web Documents," *Proc. IEEE Int'l Conf. Information Reuse Integration (IRI),* pp. 502-507, 2007.

[98] Q. Li, Y. Ding, A. Feng, and Y. Dong, "A Novel Method for Extracting Information from Web Pages with Multiple Presentation Templates," *J. Software,* vol. 5, no. 5, pp. 506-513, 2010.

[99] B. Liu and Y. Zhai, "NET: A System for Extracting Web Data from Flat and Nested Data Records," *Proc. Sixth Int'l Conf. Web Information Systems Eng. (WISE),* pp. 487-495, 2005.

[100] B. Liu, R.L. Grossman, and Y. Zhai, "Mining Web Pages for Data Records," *IEEE Intelligent Systems,* vol. 19, no. 6, pp. 49-55, Nov./Dec. 2004.

[101] K.-L. Liu, W. Meng, J. Qiu, C.T. Yu, V. Raghavan, Z. Wu, Y. Lu, H. He, and H. Zhao, "AllInOneNews: Development and Evaluation of a Large-Scale News Metasearch Engine," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 1017-1028, 2007.

[102] L. Liu, C. Pu, and W. Han, "XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources," *Proc. 16th Int'l Conf. Data Eng. (ICDE),* pp. 611-621, 2000.

[103] W. Liu, D. Shen, and T. Nie, "An Effective Method Supporting Data Extraction and Schema Recognition on the Deep Web," *Proc. 10th Asia Pacific Web Conf. (APWeb),* pp. 419-431, 2008.

[104] W. Liu, X. Meng, and W. Meng, "ViDE: A Vision-Based Approach for Deep Web Data Extraction," *IEEE Trans. Knowledge Data Eng.,* vol. 22, no. 3, pp. 447-460, Mar. 2010.

[105] Lixto, "Lixto (Home Page)," http://www.lixto.com, Jan. 2012.

[106] G.F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* third ed. Addison-Wesley, 1997.

[107] A. Machanavajjhala, A.S. Iyer, P. Bohannon, and S. Merugu, "Collective Extraction from Heterogeneous web Lists," *Proc. Fourth ACM Int'l Conf. Web Search and Data Mining (WSDM),* pp. 445-454, 2011.

[108] R. Malouf, "Markov Models for Language-Independent Named Entity Recognition," *Proc. Sixth Conf. Natural Language Learning (COLING),* pp. 1-4, 2002.

[109] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks, "Named Entity Recognition from Diverse Text Types," *Proc. Conf. Recent Advances in Natural Language Processing,* pp. 257-274, 2001.

[110] A.K. McCallum, "MALLET: A Machine Learning for Language Toolkit," technical report, UMass Amherst, 2002.

[111] G. Mecca, P. Atzeni, A. Masci, P. Merialdo, and G. Sindoni, "The Araneus Web-Based Management System," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 544-546, 1998.

[112] MEMPHIS, "Multilingual Content for Flexible Format Internet Premium Services," http://www.dfki.de/lt/project.php?id= Project_116 &l=en, 2003.

[113] W. Meng and C.T. Yu, *Advanced Metasearch Engine Technology.* Morgan & Claypool Publishers, 2010.

[114] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L.E. Moser, "Extracting Data Records from the Web using Tag Path Clustering," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 981-990, 2009.

[115] R.C. Miller and K. Bharat, "SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers," *Computer Networks,* vol. 30, no. 1-7, pp. 119-130, 1998.

[116] R.C. Miller and B.A. Myers, "Lightweight Structured Text Processing," *Proc. USENIX Ann. Technical Conf., General Track,* pp. 131-144, 1999.

[117] R.C. Miller and B.A. Myers, "LAPIS: Smart Editing with Text Structure," *Proc. Conf. Human Factors Computing Systems (CHI Extended Abstracts),* pp. 496-497, 2002.

[118] S. Miller, H. Fox, L.A. Ramshaw, and R.M. Weischedel, "A Novel Use of Statistical Parsing to Extract Information from Text," *Proc. Sixth Applied Natural Language Processing Conf. (ANLP),* pp. 226-233, 2000.

[119] S. Minton, S.I. Ticrea, and J. Beach, "Trainability: Developing a Responsive Learning System," *Proc. Workshop Information Integration Web (IIWeb),* pp. 27-32, 2003.

[120] P. Montoto, A. Pan, J. Raposo, J. Losada, F. Bellas, and V. Carneiro, "A Workflow Language for Web Automation," *J. Universal Computer Science,* vol. 14, no. 11, pp. 1838-1856, 2008.

[121] I. Muslea, "Extraction Patterns for Information Extraction Tasks: A Survey," *Proc. AAAI Workshop Machine Learning for Information Extraction,* pp. 1-6, 1999.

[122] I. Muslea, S. Minton, and C.A. Knoblock, "Hierarchical Wrapper Induction for Semistructured Information Sources," *Autonomous Agents and Multi-Agent Systems,* vol. 4, no. 1/2, pp. 93-114, 2001.

[123] A.Y. Ng, M.I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Proc. Neural Information Processing Systems (NIPS),* pp. 849-856, 2001.

[124] Online Miner, "Online Miner (home page)," http://www.temis-group.com, Jan. 2012.

[125] N. Papadakis, D. Skoutas, K. Raftopoulos, and T.A. Varvarigou, "STAVIES: A System for Information Extraction from Unknown Web Data Sources through Automatic Web Wrapper Generation using Clustering Techniques," *IEEE Trans. Knowledge Data Eng.,* vol. 17, no. 12, pp. 1638-1652, Dec. 2005.

[126] J. Park and D. Barbosa, "Adaptive Record Extraction from Web Pages," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 1335-1336, 2007.

[127] Parser Studio, "Parser Studio (Home Page)," http://www.item-field.com, Jan. 2012.

[128] L. Peshkin and A. Pfeffer, "Bayesian Information Extraction Network," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 421-426, 2003.

[129] G. Ramakrishnan, S. Joshi, S. Balakrishnan, and A. Srinivasan, "Using ILP to Construct Features for Information Extraction From Semi-Structured Text," *Proc. 17th Int'l Conf. Inductive Logic Programming (ILP),* pp. 211-224, 2007.

[130] J. Raposo, A. Pan, M. Álvarez, J. Hidalgo, and Á. Viña, "The Wargo System: Semi-Automatic Wrapper Generation in Presence of Complex Data Access Modes," *Proc. DEXA Workshops,* pp. 313-320, 2002.

[131] A. Ratnaparkhi, "Learning to Parse Natural Language with Maximum Entropy Models," *Machine Learning,* vol. 34, no. 1-3, pp. 151-175, 1999.

[132] E. Riloff, "Automatically Constructing a Dictionary for Information Extraction Tasks," *Proc. 11th Nat'l Conf. Artificial Intelligence (AAAI),* pp. 811-816, 1993.

[133] E. Riloff, "Automatically Generating Extraction Patterns from Untagged Text," *Proc. 13th Nat'l Conf. Artificial Intelligence (AAAI/IAAI),* vol. 2, pp. 1044-1049, 1996.

[134] R. Suite, "Robo Suite (Home Page)," http://kapowsoftware.com, Jan. 2012.

[135] D. Roth and W.t. Yih, "Relational Learning via Propositional Algorithms: An Information Extraction Case Study," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 1257-1263, 2001.

[136] A. Sahuguet and F. Azavant, "Building Intelligent Web Applications Using Lightweight Wrappers," *Data Knowledge Eng.,* vol. 36, no. 3, pp. 283-316, 2001.

[137] S. Sarawagi, "Information Extraction," *Foundations and Trends in Databases,* vol. 1, no. 3, pp. 261-377, 2007.

[138] K. Seymore, A. McCallum, and R. Rosenfeld, "Learning Hidden Markov Model Structure for Information Extraction," *Proc. AAAI Workshop Machine Learning for Information Extraction,* pp. 37-42, 1999.

[139] W. Shen, A. Doan, J.F. Naughton, and R. Ramakrishnan, "Declarative Information Extraction Using Datalog with Embedded Extraction Predicates," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* pp. 1033-1044, 2007.

[140] Y.K. Shen, "Automatic Record Extraction from the World Wide Web," http://dspace.mit.edu/bitstream/handle/1721.1/35609/75289843.pdf?sequence=1, 2005.

[141] Y.K. Shen and D.R. Karger, "U-REST: An Unsupervised Record Extraction System," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 1347-1348, 2007.

[142] M. Shilman, P. Liang, and P.A. Viola, "Learning Non-Generative Grammatical Models for Document Analysis," *Proc. IEEE 10th Int'l Conf. Computer Vision (ICCV),* pp. 962-969, 2005.

[143] K. Simon and G. Lausen, "ViPER: Augmenting Automatic Information Extraction with Visual perceptions," *Proc. 14th ACM Int'l Conf. Information and Knowledge Management (CIKM),* pp. 381-388, 2005.

[144] M. Skounakis, M. Craven, and S. Ray, "Hierarchical Hidden Markov Models for Information Extraction," *Proc. 18th Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 427-433, 2003.

[145] D. Smith and M. Lopez, "Information Extraction for Semi-Structured Documents," *Proc. Workshop Management Semistructured Data,* 1997.

[146] S. Soderland, "Learning to Extract Text-Based Information from the World Wide Web," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 251-254, 1997.

[147] S. Soderland, "Learning Information Extraction Rules for Semi-Structured and Free Text," *Machine Learning,* vol. 34, no. 1-3, pp. 233-272, 1999.

[148] S. Soderland, D. Fisher, J. Aseltine, and W.G. Lehnert, "CRYSTAL: Inducing a Conceptual Dictionary," *Proc. 14th Int'l Joint Conf. Artificial Intelligence (IJCAI),* pp. 1314-1321, 1995.

[149] W. Su, J. Wang, and F.H. Lochovsky, "ODE: Ontology-Assisted Data Extraction," *ACM Trans. Database Systems,* vol. 34, no. 2, article 12, 2009.

[150] A. Sun, M.-M. Naing, E.-P. Lim, and W. Lam, "Using Support Vector Machines for Terrorism Information Extraction," *Proc. First NSF/NIJ Conf. Intelligence and Security Informatics (ISI),* pp. 1-12, 2003.

[151] C. Tao and D.W. Embley, "Automatic Hidden-Web Table Interpretation, Conceptualization, and Semantic Annotation." *Data Knowledge Eng.,* vol. 68, no. 7, pp. 683-703, 2009.

[152] J. Tatemura, S. Chen, F. Liao, O. Po, K. Selçuk Candan, and D. Agrawal, "UQBE: Uncertain Query by Example for Web Service Mashup," *Proc. ACM SIGMOD Int'l Conf. Management of Data,,* pp. 1275-1280, 2008.

[153] Text Pipe, "Text Pipe (Home Page)," http://www.datamystic.com, Jan. 2012.

[154] The .GOV Test Collection, "The .GOV Test Collection," http://ir.dcs.gla.ac.uk/test_collections/govinfo.html, Jan. 2012.

[155] B. Thomas, "Anti-Unification Based Learning of T-Wrappers for Information Extraction," *Proc. AAAI Workshop Machine Learning for Information Extraction,* pp. 15-20, 1999.

[156] J. Turmo and H. Rodríguez, "Learning Rules for Information Extraction," *Natural Language Eng.,* vol. 8, pp. 167-191, 2002.

[157] J. Turmo, A. Ageno, and N. Català, "Adaptive Information Extraction," *ACM Computing Surveys,* vol. 38, no. 2, article 4, 2006.

[158] U. Studio, "Uno Studio (Home Page)," http://www.orsus.com, Jan. 2012.

[159] Unwwwrap, "Unwwwrap (Home Page)," http://www.extradata.com, Jan. 2012.

[160] P.A. Viola and M. Narasimhan, "Learning to Extract Information from Semi-Structured Text Using a Discriminative Context Free Grammar," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR),* pp. 330-337, 2005.

[161] Visual Web Task, "Visual Web Task (Home Page)," http://www.lencom.com, Jan. 2012.

[162] vTag, "vTag (Home Page)," http://www.connotate.com, Jan. 2012.

[163] J. Wang and F.H. Lochovsky, "Data-Rich Section Extraction from HTML Pages," *Proc. Third Int'l Conf. Web Information Systems Eng. (WISE),* pp. 313-322, 2002.

[164] J. Wang and F.H. Lochovsky, "Data Extraction and Label Assignment for Web Databases," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 187-196, 2003.

[165] Web Activity, "Web Activity (Home Page)," http://www.knowmadic.com, Jan. 2012.

[166] Web Data Kit, "Web Data Kit (Home Page)," http://www.lotontech.com, Jan. 2012.

[167] Webinator, "Webinator (Home Page)," http://www.thunderstone.com, Jan. 2012.

[168] WebQL, "WebQL (Home Page)," http://www.ql2.com, Jan. 2012.

[169] L. Wei, X. Meng, and W. Meng, "Vision-Based Web Data Records Extraction," *Proc. Int'l Workshop Web and Databases (WebDB),* 2006.

[170] F. Wu and D.S. Weld, "Autonomously Semantifying Wikipedia," *Proc. ACM Conf. Information and Knowledge Management (CIKM),* pp. 41-50, 2007.

[171] X-Fetch Wrapper, "X-Fetch Wrapper," http://www.x-fetch.com, Jan. 2012.

[172] XRover, "XRover (Home Page)," http://www.xsb.com, Jan. 2012.

[173] Y. Yamada, N. Craswell, T. Nakatoh, and S. Hirokawa, "Testbed for Information Extraction from the Deep Web," *Proc. 13th Int'l World Wide Web Conf. (WWW) (Alternate Track Papers & Posters),* pp. 346-347, 2004.

[174] R. Yangarber, "Counter-Training in Discovery of Semantic Patterns," *Proc. 41st Ann. Meeting. Assoc. Computational Linguistics (ACL),* pp. 343-350, 2003.

[175] L. Yi, B. Liu, and X. Li, "Eliminating Noisy Information in Web Pages for Data Mining," *Proc. ACM SIGKDD Ninth Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 296-305, 2003.
[176] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma, "Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation," *Proc. Int'l Conf. World Wide Web (WWW)*, pp. 11-18, 2003.
[177] D. Zelenko, C. Aone, and A. Richardella, "Kernel Methods for Relation Extraction." *J. Machine Learning Research*, vol. 3, pp. 1083-1106, 2003.
[178] Y. Zhai and B. Liu, "Structured Data Extraction from the Web based on Partial Tree Alignment," *IEEE Trans. Knowledge Data Eng.*, vol. 18, no. 12, pp. 1614-1628, Dec. 2006.
[179] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C.T. Yu, "Fully Automatic Wrapper Generation for Search Engines," *Proc. Int'l Conf. World Wide Web (WWW)*, pp. 66-75, 2005.
[180] H. Zhao, W. Meng, and C.T. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," *Proc. 32nd Int'l Conf. Very Large Datya Bases (VLDB)*, pp. 989-1000, 2006.
[181] S. Zhao and R. Grishman, "Extracting Relations with Integrated Information Using Kernel Methods," *Proc. 43rd Ann. Meeting Assoc. Computational Linguistics (ACL)*, 2005.
[182] J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma, "Simultaneous Record Detection and Attribute Labeling in Web Data Extraction," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 494-503, 2006.

Hassan A. Sleiman received the MSc degree from the University of Sevilla in 2008. Currently, he is a lecturer in the Department of Computer Languages and Systems at the University of Sevilla, Spain. He was a software engineer for companies such as Dynagent and Indevia Solutions, and a researcher for The Distributed Group (TDG), where he became involved in several projects on enterprise information integration and enterprise application integration. His current research interests include web information extraction, as well as enterprise information integration.

Rafael Corchuelo received the PhD degree from the University of Seville, Spain, and he has led its Research Group on Distributed Systems (TDG) since 1997. He is a reader of software engineering who is with the Department of Computer Languages and Systems of the University of Seville, Spain. His current research interests focus on the integration of web data islands; previously, he was on multiparty interaction and fairness issues.