A Survey of Web Information Extraction Systems

Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, Khaled Shaalan

Abstract—The Internet presents a huge amount of useful information which is usually formatted for its users, which makes it difficult to extract relevant data from various sources. Therefore, the availability of robust, flexible Information Extraction (IE) systems that transform the Web pages into program-friendly structures such as a relational database will become a great necessity. Although many approaches for data extraction from Web pages have been developed, there has been limited effort to compare such tools. Unfortunately, in only a few cases can the results generated by distinct tools be directly compared since the addressed extraction tasks are different. This paper surveys the major Web data extraction approaches and compares them in three dimensions: the task domain, the automation degree, and the techniques used. The criteria of the first dimension explain why an IE system fails to handle some Web sites of particular structures. The criteria of the second dimension classify IE systems based on the techniques used. The criteria of the third dimension measure the degree of automation for IE systems. We believe these criteria provide qualitatively measures to evaluate various IE approaches.

Index Terms—Information Extraction, Web Mining, Wrapper, Wrapper Induction.

1 Introduction

THE explosive growth and popularity of the world-wide web has resulted in a huge amount of information sources on the Internet. However, due to the heterogeneity and the lack of structure of Web information sources, access to this huge collection of information has been limited to browsing and searching. Sophisticated Web mining applications, such as comparison shopping robots, require expensive maintenance to deal with different data formats. To automate the translation of input pages into structured data, a lot of efforts have been devoted in the area of information extraction (IE). Unlike information retrieval (IR), which concerns how to identify relevant documents from a document collection, IE produces structured data ready for post-processing, which is crucial to many applications of Web mining and searching tools.

Formally, an IE task is defined by its input and its extraction target. The input can be unstructured documents like free text that are written in natural language (e.g. Figure 1) or the semi-structured documents that are pervasive on the Web, such as tables or itemized and enumerated lists (e.g. Figure 2). The extraction target of an IE task can be a relation of k-tuple (where k is the number of attributes in a record) or it can be a complex object with hierarchically organized data. For some IE tasks, an attribute may have zero (missing) or multiple instantiations in a record. The difficulty of an IE task can be further complicated when various permutations of attributes or typographical errors occur in the input documents.

Programs that perform the task of IE are referred to as extractors or wrappers. A wrapper was originally defined as a component in an information integration system which aims at providing a single uniform query interface to access multiple information sources. In an information integration system, a wrapper is generally a program that "wraps" an information source (e.g. a database server, or a Web server) such that the information integration system can access that information source without changing its core query answering mechanism. In the case where the information source is a Web server, a wrapper must query the Web server to collect the resulting pages via HTTP protocols, perform information extraction to extract the contents in the HTML documents, and finally integrate with other data sources. Among the three procedures, information extraction has received most attentions and some use wrappers to denote extractor programs. Therefore, we use the terms extractors and wrappers interchangeably.

Wrapper induction (WI) or information extraction (IE) systems are software tools that are designed to generate wrappers. A wrapper usually performs a pattern matching procedure (e.g., a form of finite-state machines) which relies on a set of extraction rules. Tailoring a WI system to a new requirement is a task that varies in scale depending on the text type, domain, and scenario. To maximize reusability and minimize maintenance cost, designing a trainable WI system has been an important topic in the research fields of message understanding, machine learning, data mining, etc. The task of Web IE, that we are concerned in this paper, differs largely from traditional IE tasks in that traditional IE aims at extracting data from totally unstructured free texts that are written in natural language. Web IE, in contrast, processes online documents that are semi-structured and usually generated automatically by a server-side application program. As a result, traditional IE usually takes advantage of NLP techniques such as lexicons and grammars, whereas Web IE usually applies machine learning and pat-

- *Chia-Hui Chang is with the Department of Computer Science and Information Engineering, National Central University, No. 300, Jungda Rd., Jhongli City, Taoyuan, Taiwan 320, R.O.C., E-mail: chia@csie.ncu.edu.tw.*
- *Mohammed Kayed is with the Mathematics Department, Beni-Suef University, Egypt, E-mail: mskayed@yahoo.com.*
- *Moheb Ramzy Girgis is with the Department of Computer Science, Minia University, El-Minia, Egypt, E-mail: mrgirgis@mailer.eun.eg.*
- *Khaled Shaalan is with The British University in Dubai (BUiD), United Arab Emirates, E-mail: khaled.shaalan@buid.ac.ae.*

Manuscript received (insert date of submission if desired). Please note that all acknowledgments should be placed at the end of the paper, before the bibliography.

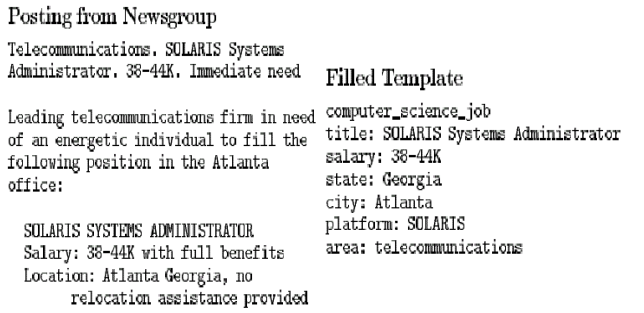

Fig. 1. A free text IE task which is specified by the input (left) and its output (right).

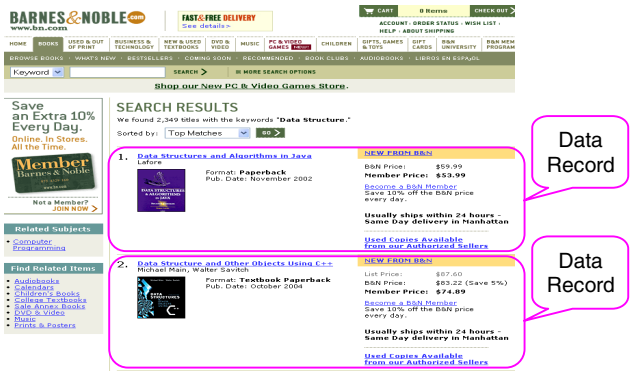

Fig. 2. A Semi-structured page containing data records (in rectangular box) to be extracted.

tern mining techniques to exploit the syntactical patterns or layout structures of the template-based documents.

In this paper, we focus on IE from semi-structured documents and discuss only those that have been used for Web data. We will compare different WI systems using features from three dimensions which we regard as criteria for comparing and evaluating WI systems. The rest of the paper is organized as follows. Section 2 introduces related work on WI system taxonomy, which we summarize into three dimensions of evaluating WI systems. Section 3 suggests the criteria for each dimension. We make a survey of contemporary IE tools in Section 4 with a running example to make such tools more understandable. A comparative analysis of the surveyed IE tools from the three dimensions is presented in Section 5. Finally, the conclusions are made in Section 6.

2 Related Work

In the past few years, many approaches to WI systems, including machine learning and pattern mining techniques, have been proposed, with various degrees of automation. In this section we survey the previously proposed taxonomies for IE tools developed by the main researchers.

The Message Understanding Conferences (MUCs) have inspired the early work in IE. There are five main tasks defined for text IE, including named entity recognition, coreference resolution, template element construction, template relation construction and scenario template production. The significance of the MUCs in the field of IE motivates some researchers to classify IE approaches into two different classes: MUC Approaches (e.g., AutoSolg [1], LIEP [2], PALKA [3], HASTEN [4], and CRYSTAL [5]) and Post-MUC Approaches (e.g., WHISK [6], RAPIER [7], SRV [8], WIEN [9], SoftMealy [10] and STALKER [11]).

Hsu and Dung [10] classified wrappers into 4 distinct categories, including hand-crafted wrappers using general programming languages, specially designed programming languages or tools, heuristic-based wrappers, and WI approaches. Chang [12] followed this taxonomy and compared WI systems from the user point of view and discriminated IE tools based on the degree of automation. They classified IE tools into four distinct categories, including systems that need programmers, systems that need annotation examples, annotation-free systems and semi-supervised systems.

Muslea, who maintains the RISE (Repository of Online Information Sources Used in Information Extraction Tasks) Web site, classified IE tools into 3 different classes according to the type of input documents and the structure/constraints of the extraction patterns [11]. The first class includes tools that process IE from free text using extraction patterns that are mainly based on syntactic/semantic constraints. The second class is called Wrapper induction systems which rely on the use of delimiter-based rules since the IE task processes online documents such as HTML pages. Finally, the third class also processes IE from online documents; however the patterns of these tools are based on both delimiters and syntactic/semantic constraints.

Kushmerick classified many of the IE tools into two distinct categories finite-state and relational learning tools [13]. The extraction rules in finite-state tools are formally equivalent to regular grammars or automata, e.g WIEN, SoftMealy and STALKER, while the extraction rules in relational learning tools are essentially in the form of Prolog-like logic programs, e.g. SRV, Crystal, WebFoot [14], Rapier and Pinocchio [15].

Laender proposed a taxonomy for data extraction tools based on the main technique used by each tool to generate a wrapper [16]. These include languages for wrapper development (e.g., Minerva [17], TSIMMIS [18] and WebOQL [19]), HTML-aware tools (e.g., W4F [20], XWrap [21] and RoadRunner [22]), NLP-based tools (e.g., WHISK, RAPIER and SRV), Wrapper induction tools (e.g., WIEN, SoftMealy and STALKER), Modeling-based tools (e.g., NoDoSE [23] and DEByE [24],[25], and Ontology-based tools (e.g., BYU [26]). Laender compared among the tools by using the following 7 features: degree of automation, support for complex objects, page contents, availability of a GUI, XML output, support for non-HTML sources, resilience, and adaptiveness.

Sarawagi classified HTML wrappers into 3 categories according to the kind of extraction tasks [27]. The first category, record-level wrappers, exploits regularities to discover record boundaries and then extract elements of a single list of homogeneous records from a page. The second category, page-level wrappers, extracts elements of multiple kinds of records. Finally, the site-level wrappers populate a database from pages of a Web site.

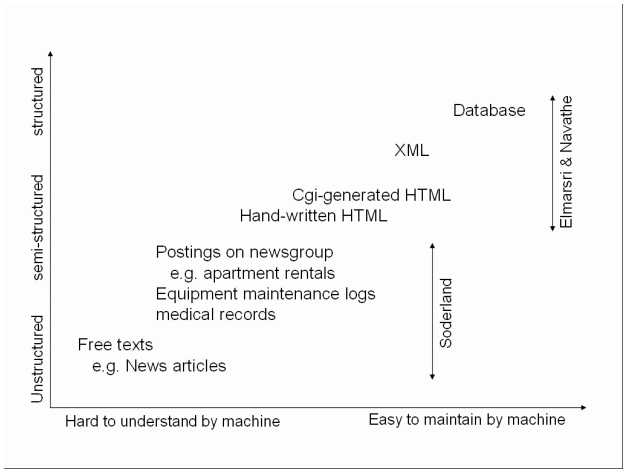

Fig. 3. Structurization of various documents.

Kuhlins and Tredwell classified the toolkits for generating wrappers into two basic categories, based on commercial and non-commercial availability [28]. They also contrasted the toolkits by using some features such as output methods, interface type, web crawling capability and GUI support.

This survey shows three main dimensions for evaluating IE systems. First, the distinction of free text IE and online documents made by Muslea, the three-level of extraction tasks proposed by Sarawagi, and the capabilities of handling non-HTML sources, together suggest the first dimension, which concerns the difficulty or the task domain that an IE task refers to. Second, the taxonomy of regular expression rules or Prolog-like logic rules, and that of deterministic finite-state transducer or probabilistic hidden Markov models, prompts the second dimension which relates the underlying techniques used in IE systems. Finally, the categorizations of programmer-involved, learning-based or annotation-free approaches imply the third dimension which concerns the degree of automation. These three dimensions are discussed in the next section.

3 Three Dimensions for Comparing IE Systems

Continuing our survey of various taxonomies, there are three dimensions to be used in the comparison. The first dimension evaluates the difficulty of an IE task, which can be used to answer the question "why an IE system fails to handle some Web sites with particular structures?" The second dimension compares the techniques used in different IE systems. The third dimension evaluates both the effort made by the user for the training process and the necessity to port an IE system across different domains. From the user's point of view, the second dimension is less important. However, researchers might get an overview of which machine-learning or data mining technologies have been used for WI through the comparison. In this section we describe each of these dimensions, and for each one we include a set of features that can be used as criteria for comparing and evaluating IE systems from the dimension prospective.

3.1 Task difficulties

The input file of an IE task may be structured, semi-structured or free-text. As shown in Figure 3, the definition of these terms varies across research domains. Soderland [14] considered free-texts e.g. news article, that are written in natural languages are unstructured, postings on newsgroup (e.g. apartment rentals), medical records, equipment maintenance logs are semi-structured, while HTML pages are structured. However, from the viewpoint of database researchers [29], the information stored in databases is known as structured data; XML documents are semi-structured data for the schema information is mixed in with the data values, while Web pages in HTML are unstructured because there is very limited indication of the type of data. From our viewpoints, XML documents are considered as structured since there are DTD or XML schema available to describe the data. Free texts are unstructured since they require substantial natural language processing. For the large volume of HTML pages on the Web, they are considered as semi-structured [10] since the embedded data are often rendered regularly via the use of HTML tags.

Thus, semi-structured inputs are the documents of a fairly regular structure and data in them may be presented in HTML or non-HTML format. One source of these large semi-structured documents is from the deep Web, which includes dynamic Web pages that are generated from structured databases with some templates or layouts. For example, the set of book pages from Amazon has the same layout for the authors, title, price, comments, etc. Web pages that are generated from the same database with the same template (program) form a page class. There are also semi-structured HTML pages generated by hand. For example, the publication lists from various researchers' homepages all have title and source for each single paper, though they are produced by different people. For many IE tasks, the input are pages of the same class, still some IE tasks focus on information extraction from pages across various Web sites.

In addition to the categorization by input documents, an IE task can be classified according to the extraction target. For example, Sarawagi classified HTML wrappers into record-level, page-level and site-level IE tasks. Record-level wrappers discover record boundaries and then divide them into separate attributes; page-level wrappers extract all data that are embedded in one Web page, while site-level wrappers populate a database from pages of a Web site, thus the attributes of an extraction object are scattered across pages of a Web site. Academic researchers have devoted much effort to develop record-level and page-level data extraction, whereas industrial researchers have more interest in complete suites which support site-level data extraction.

There are various ways to describe the extraction targets in a page. The most common structure (as proposed in NoDoSE, DEByE, and Stalker, etc.) is a hierarchical tree where the leaf nodes are *basic* types while the internal nodes are *list of types*. A data object may be of a plain/nested structure. A plain text data-object has only one internal node (the root), while a nested data-object contains more than two levels and internal nodes. Since these Web pages are intended to be human readable, tuples of the

same list, or elements of a tuple are often expressly separated or delimited for easy visualization. However, the presentation formats or the set of attributes that form a data-object is subject to the following variations:

- An attribute may have zero or more values (list of 1-tuple) in a data-object. If the attribute has zero value, it is called a *missing* attribute; if it has more than one value, it is called a *multi-valued* attribute. The name of a book's author may be an example of multi-valued attribute, whereas a special offer, which is available only for certain books, is an example of missing attribute.
- The set of attributes (A_1, A_2, …, A_k) may have multiple ordering, i.e., an attribute A_i may have variant positions in different instances of a data-object; and we call this attribute a *multi-ordering* attribute. For example, a movie site might list the release date before the title for movies prior to 1999, but after the title for recent movies.
- An attribute may have variant formats along with different instances of a data-object. If the format of an attribute is not fixed, we might need disjunctive rules to generalize all cases. For example, an e-commerce site might list prices in bold face, except for sale prices which are in red. So, price would be an example of a variant-format attribute in this site. On the other hand, different attributes in a data-object may have the same format, especially in table presentation, where single <TD> tags are used to present various attributes. In such cases, order of attributes is the key information to distinguish various attributes. However, if missing attributes occur or multi-ordering exists, the extraction rules for these attributes need to be revised.
- Most IE systems handle input documents as strings of tokens for they are easier to process than strings of characters. Depending on the tokenization methods used, sometimes an attribute can not be decomposed into individual tokens. Such an attribute is called an *untokenized* attribute. For example, in a college course catalogue the department code has no delimiter separated it from the course number in strings such as "COMP4016" or "GEOL2001". The granularity of extraction targets affects the decision/selections of tokenization schemes for an IE system.

The combination of various input documents and variation of extraction targets causes different degrees of task difficulties. Since various IE systems are designed for various IE tasks, it is not fair to compare them directly. However, analyzing what task an IE system targets and how it accomplishes the task, can be used to evaluate this system and possibly extend to other task domains.

3.2 The Techniques Used

For a wrapper to extract data from input it needs to tokenize the input string, apply the extraction rules for each attribute, assemble the extracted values into records, and repeat the process for all object instances in the input. There are various granularities for input string tokenization, including tag-level and word-level encoding. The former encoding translates each HTML tag as a token and translates any text string between two tags as a special token, while the later, word-level, treats each word in a document as a token. Extraction rules can be induced by top-down or bottom-up generalization, pattern mining, or logic programming. The type of extraction rules may be expressed using regular grammars or logic rules. Some of the WI systems use path-expressions of the HTML parse tree path (e.g. html.head.title, and html->table[0]) as the features in extraction rules; some use syntactic or semantic constraints, such as POS-tags and WordNet semantic class; while others use delimiter-based constraints, such as HTML tags or literal words, in the extraction rules. The extractor architecture may require single or multiple passes over the pages.

In summary, the features for comparing WI systems from the perspective of techniques used include: *tokenization/encoding schemes*, *scan pass*, *extraction rule type*, *features involved*, and *learning algorithm*.

3.3 Automation Degree

As described above, a wrapper program has many phases to be accomplished: collecting training pages, labeling training examples, generalizing extraction rules, extracting the relevant data, and outputting the result in an appropriate format. Most researches focus on the intermediate 3 phases which involve the major extraction process, while some provide a total solution including a crawler or robot for collecting training pages (the first phase) and an output support in XML format or back-end relational database for further information integration (the final phase). Generally speaking, the labeling phase defines/specifies the output of an extraction task and requires the involvement of users. However, some WI systems do not require the collected training examples to be labeled before the learning stage, instead, the labeling or annotation of the extracted data can be done after the generation of extraction rules (with or without users). This brings up a major difference in automation: for some WI systems, the user needs to label training examples; for others, the user simply waits for the systems to clean the pages and extract the data. However, the automation does not come without reason. The cost is the applicability of these approaches to other task domain. Some even have limitation in the number and the type of input pages.

In summary, the features we consider from the automation degree prospective include: *user expertise* needed for labeling data or for generating the extraction rules, *applicability* of these approaches to other task domain, *limitation* for the number/type of input, *page-fetching support* for collecting training pages, *output support* and *API support* for application integration.

4 Survey for Contemporary IE Systems

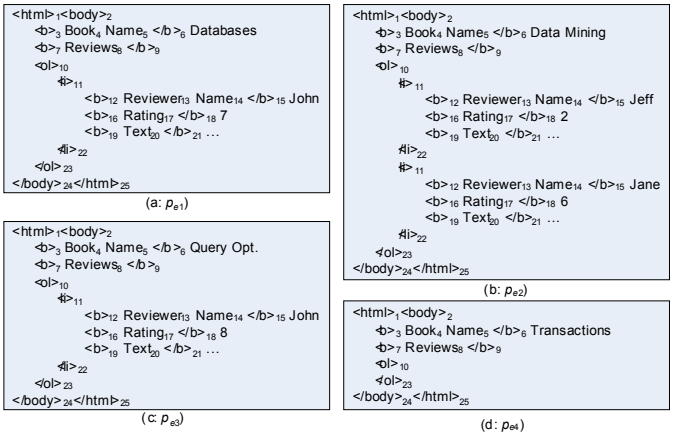

Fig. 4. A running example of four Web pages (p_{e1}-p_{e4}).

The goal of WI is to automatically generate a wrapper that is used to extract the targets for an information resource. Let us consider the way how user interacts with WI systems. Earlier systems are designed to facilitate programmers in writing extraction rules, while later systems introduce machine learning for automatic rule generalization. Therefore, the user interaction has evolved from writing extraction rules to labeling target extraction data. In recent years, more efforts are devoted to reducing labeling and creating WI systems with unlabelled training examples. Following this trend, we can classify WI systems into the four classes *manually-constructed* IE Systems, *supervised* IE Systems, *semi-supervised* IE Systems and *unsupervised* IE Systems.

In this section we give a survey for most prominent and contemporary IE approaches. To make such approaches more understandable, we assume an IE task and describe the generated wrapper that can be used to extract information from other similar documents for each approach. Figure 4 shows four Web pages as the input of the IE task. The desired output is the book title and the corresponding reviews, including the reviewer name, rating and comments.

4.1 Manually-constructed IE systems

As shown on the right of Figure 5, in manually-constructed IE systems, users program a wrapper for each Web site by hand using general programming languages such as Perl or by using special-designed languages. These tools require the user to have substantial computer and programming backgrounds, so it becomes expensive. Such systems include TSIMMIS, Minerva, Web-OQL, W4F and XWRAP.

TSIMMIS is one of the first approaches that give a framework for manual building of Web wrappers [18]. The main component of this project is a wrapper that takes as input a specification file that declaratively states (by a sequence of commands given by programmers) where the data of interest is located on the pages and how the data should be "packaged" into objects. For example, Figure 6(a) shows the specification file for our IE task in Figure 4. Each command is of the form: [*variables*, *source*, *pattern*], where *source* specifies the input text to be considered, *pattern* specifies how to find the text of interest within the source, and *variables* are a list of variables that hold the extracted results. The special

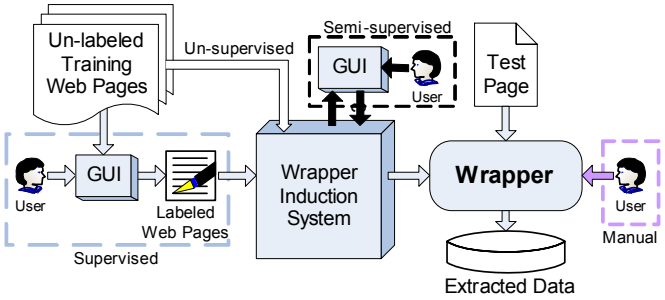

Fig. 5. A general view of WI systems.

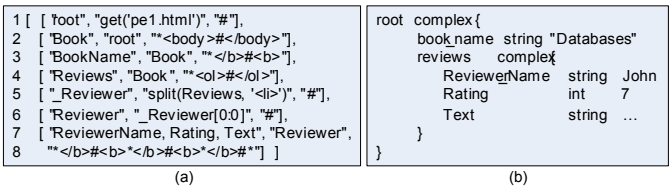

Fig. 6. (a) A TSIMMIS specification file and (b) the OEM output.

symbol '*' in a pattern means discard, and '#' means save in the variables. TSIMMIS then outputs data in Object Exchange Model (e.g. Figure 6(b)) that contains the desired data together with information about the structure and the contents of the result. TSIMMIS provides two important operators: *split* and *case*. The split operator is used to divide the input list element into individual elements (e.g. line 5). The case operator allows the user to handle the irregularities in the structure of the input pages.

Minerva attempts to combine the advantages of a declarative grammar-based approach with the flexibility of procedural programming in handling heterogeneities and exceptions [17]. This is done by incorporating an explicit exception-handling mechanism inside a regular grammar. Exception-handling procedures are written in Minerva by using a special language called Editor. The grammar used by Minerva is defined in an EBNF style where a set of productions is defined; each production rule defines the structure of a non-terminal symbol (preceded by '$') of the grammar. For example, Figure 7 shows the set of productions that can be used to extract (also, insert in a database) relevant attributes for the defined IE task. As usual in EBNF notation, expression [*p*] denotes an optional pattern *p*; expression (*p*)* denotes that *p* may be repeated zero or more times. The nonterminal productions $bName, $rName, $rate, and $text immediately follow from their use in the definition of $Book. Thus, book name is preceded by "Book Name" and followed by "" as indicated by pattern "*(?)" which matches every thing before tag . The last production in Figure 7 defines a special non-terminal $TP (Tuple Production), which is used to insert a tuple in the database after each book has been parsed. For each production rule, it is possible to add an exception handler containing a piece of Editor code that can handles the irregularities found in the Web data. Whenever the parsing of that production rule fails, an exception is raised and the corre-

```
Page Book_Reviews
$Book_Reviews: <html><body> $Book </body></html>;
$Book:   <b>Book Name </b> $bName <b> Reviews </b>
         [<ol> ( <li><b> Reviewer Name </b> $rName <b>
                 Rating </b>$rate <b> Text </b> $text $TP  )* </ol>];
$bName:                  *(?<b>);
$rName:                  *(?<b>);
$rate:                   *(?<b>);
$text:                   *(?</li>);

$TP:        {
                         $bName, $rName
                         $rate
                         $text
            }
END
```

Fig. 7. A Minerva grammar in ENBF style.

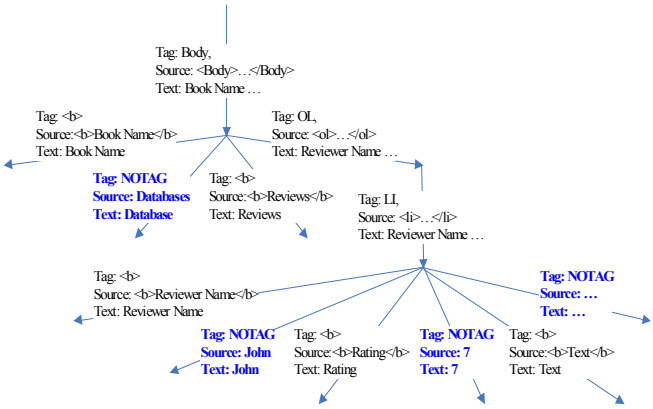

Fig. 8. A WebOQL hypertree for the page p_{e1} in Figure 4.

sponding exception handler is executed.

WebOQL is a functional language that can be used as query language for the Web, for semistructured data and for website restructuring [19]. The main data structure provided by WebOQL is the *hypertree*. Hypertrees are arc-labeled ordered trees which can be used to model a relational table, a Bibtex file, a directory hierarchy, etc. The abstraction level of the data model is suitable to support collections, nesting, and ordering. Figure 8 shows the hypertree for page p_{e1} of the running example. As shown in the figure, the tree structure is similar to the DOM tree structure where arcs are labeled with records with three attributes Tag, Source, Text, corresponding to tag name, the piece of HTML code, and the text excluding markup, respectively. The main construct provided by WebOQL is the familiar select-from-where. The language has the ability to simulate all operations in nested relational algebra and compute transitive closure on an arbitrary binary relation. As an example, the following query extracts the reviewer names "Jeff" and "Jane" from page p_{e2}, where quote and exclamation mark denote the first subtree and the tail tree, respectively. The variables, depending on the case, iterate over the simple trees or tail trees of the hypertree specified after operator "in".

Select [Z!'.Text]
From *x* in **browse** ("pe2.html")', *y* in *x*', Z in *y*'
Where *x*.Tag = "ol" and Z.Text="Reviewer Name"

In addition to manage the data using the hypertrees, the language can also be used to Web restructuring making the query result readable for other applications.

W4F (Wysiwyg Web Wrapper Factory) is a Java toolkit to generate Web wrappers [20]. The wrapper development process consists of three independent layers: *retrieval*, *extraction* and *mapping* layers. In the retrieval layer, a to-be-processed document is retrieved (from the Web through HTTP protocol), cleaned and then fed to an HTML parser that constructs a parse tree following the Document Object Model (DOM). In the extraction layer, extraction rules are applied on the parse tree to extract information and then store them into the W4F internal format called Nested String List (NSL). In the mapping layer, the NSL structures are exported to the upper-level application according to mapping rules. Extraction rules are expressed using the HEL (HTML Extraction Language), which uses the HTML parse tree (i.e. DOM tree) path to address the data to be located. For example, to address the reviewer's name "Jeff" and "Jane" from p_{e2}, we can use expression <<html.body.ol[0].li[*].pcdata[0].txt>> where the symbol [*] can match any number (in this case, 0 and 1). The language also offers regular expressions and constraints to address finer pieces of data. For example, users can use regular expression to *match* or *split* (following the Perl syntax) the string obtained by DOM tree path. Finally, the *fork* operator allows the construction of nested string list by following multiple sub-paths at the same time. To assist the user addressing DOM tree path, the toolkit is designed with wysiwyg (what you see is what you get) support via smart wizards.

XWrap is a system that exploits formatting information in Web pages to hypothesize the underlying semantic structure of a page [21]. It encodes the hypothetical structure and the extraction knowledge of the web pages in a rule-based declarative language designed specifically for XWrap. The wrapper generation process includes two phases: *structure analysis*, and *source-specific XML generation*. In the first phase, XWrap fetches, cleans up, and generates a tree-like structure for the page. The system then identifies regions, semantic tokens of interest and useful hierarchical structures of sections of the page by interacting with users through object (record) and element extraction steps. In the second phase, the system generates a XML template file based on the content tokens and the nesting hierarchy specification, and then constructs a source-specific XML generator. In a way, XWRap can be classified as supervised WI systems for no rule writing is necessary; however, it requires users' understanding of the HTML parse tree, the identification of the separating tags for rows and columns in a table, etc. Thus, it is classified as systems that require special expertise of users. On the other hand, no specific learning algorithm is used here; the extraction rules are mainly based on DOM-tree path addressing.

4.2 Supervised WI systems

As shown in the left-bottom of Figure 5, supervised WI systems take a set of web pages labeled with examples of the data to be extracted and output a wrapper. The user provides an initial set of labeled examples and the system (with a GUI) may suggest additional pages for the user to label. For such systems, general users instead of program-

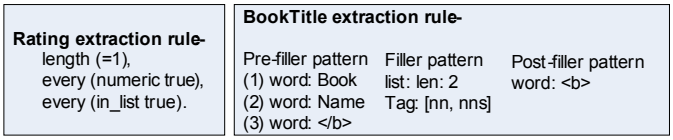

Fig. 9. A SRV (a) and Rapier (b) extraction rules.

mers can be trained to use the labeling GUI, thus reducing the cost of wrapper generation. Such systems are SRV, RAPIER, WHISK, WIEN, STALKER, SoftMealy, NoDoSE and DEByE.

SRV is a top-down relational algorithm that generates single-slot extraction rules [8]. It regards IE as a kind of classification problem. The input documents are tokenized and all substrings of continuous tokens (i.e. text fragments) are labeled as either extraction target (positive examples) or not (negative examples). The rules generated by SRV are logic rules that rely on a set of token-oriented features (or predicates). These features have two basic varieties: simple and relational. A simple feature is a function that maps a token into some discrete value such as length, character type (e.g., numeric), orthography (e.g., capitalized) and part of speech (e.g., verb). A relational feature maps a token to another token, e.g. the contextual (previous or next) tokens of the input tokens. The learning algorithm proceeds as FOIL, starting with entire set of examples and adds predicates greedily to cover as many positive examples and as few negative examples as possible. For example, to extract the rating score for our running example, SRV might return rule like Figure 9(a), which says rating is a single numeric word and occurs within a HTML list tag.

RAPIER also focuses on field-level extraction but uses bottom-up (compression-based) relational learning algorithm [7], i.e. it begins with the most specific rules and then replacing them with more general rules. RAPIER learns single slot extraction patterns that make use of syntactic and semantic information including part-of-speech tagger or a lexicon (WordNet). The extraction rules consist of three distinct patterns. The first one is the pre-filler pattern that matches text immediately preceding the filler, the second one is the pattern that match the actual slot filler, finally the last one is the post-filler pattern that match the text immediately following the filler. As an example, Figure 9(b) shows the extraction rule for the book title, which is immediately preceded by words "Book", "Name", and "", and immediately followed by the word "". The "Filler pattern" specifies that the title consists of at most two words that were labeled as "nn" or "nns" by the POS tagger (i.e., one or two singular or plural common nouns).

WIEN: Kushmerick identified a family of six wrapper classes, LR, HLRT, OCLR, HOCLRT, N-LR and N-HLRT for semi-structured Web data extraction [9]. WIEN focuses on extractor architectures. The first four wrappers are used for semi-structured documents, while the remaining two wrappers are used for hierarchically nested documents. The LR wrapper is a vector of 2*K* delimiters for a site containing

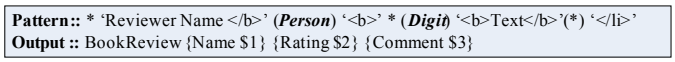

Fig. 10. A WHISK extraction rule.

K attributes. For example, the vector ('Reviewer name ', '', 'Rating ', '', 'Text ', '
') can be used to extract 3-slot book reviews for our running example. The HLRT class uses two additional delimiters to skip over potentially-confusing text in either the head or tail of the page. The OCLR class uses two additional delimiters to identify an entire tuple in the document, and then uses the LR strategy to extract each attribute in turn. The HOCLRT wrapper combines the two classes OCLR and HLRT. The two wrappers N-LR and N-HLRT are extension of LR and HLRT and designed specifically for nested data extraction. Note that, since WIEN assumes ordered attributes in a data record, missing attributes and permutation of attributes can not be handled.

WHISK uses a covering learning algorithm to generate multi-slot extraction rules for a wide variety of documents ranging from structured to free text [6]. When applying to free text, WHISK works best with input that has been annotated by a syntactic analyzer and a semantic tagger. WHISK rules are based on a form of regular expression patterns that identify the context of relevant phrases and the exact delimiters of those phrases. It takes a set of hand-tagged training instances to guide the creation of rules and to test the performance of the proposed rules. WHISK induces rules top-down, starting from the most general rule that covers all instances, and then extending the rule by adding terms one at a time. For example, to generate 3-slot book reviews, it start with empty rule "*(*)*(*)*(*)*", where each parenthesis indicates a phrase to be extracted. The phrase within the first set of parentheses is bound to the first variable $1, and the second to $2, and forth. Thus, the rule in Figure 10 can be used to extract our 3-slot book reviews for our running example. If part of the input remains after the rule has succeeded, the rule is re-applied to the rest of the input. Thus, the extraction logic is similar to the LR wrapper for WIEN.

NoDoSE: Opposed to WIEN, where training examples are obtained from some oracles that can identify interesting types of fields within a document, NoDoSE provides an interactive tool for users to hierarchically decompose semi-structured documents (including plain text or HTML pages) [23]. Thus, NoDoSE is able to handle nested objects. The system attempts to infer the format/grammar of the input documents by two heuristic-based mining components: one that mines text files and the other parses HTML code. Similar to WIEN, the mining algorithms try to find common prefix and suffix as delimiters for various attributes. Although it does not assume the order of attributes within a record to be fixed, it seeks to find a totally consistent ordering for various attributes in a record. The result of this task is a tree that describes the structure of the document. For example, to generate a wrapper for the running example, the user can interact with the NoDoSE GUI to decompose the document as a record with two fields: a book title (an

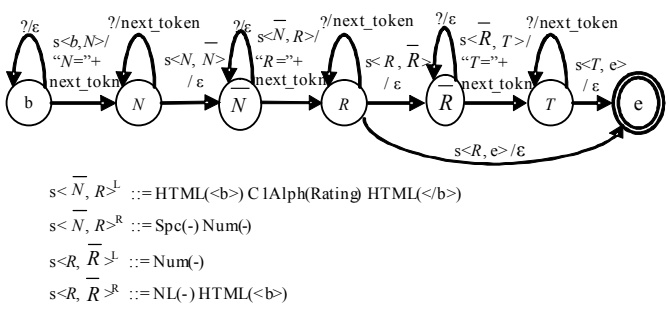

Fig. 11. A FST for the Web pages in the running example.

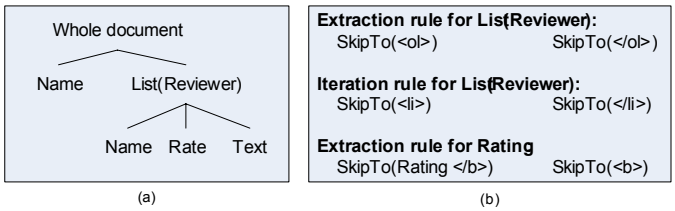

Fig. 12. An EC tree (a), and a Stalker extraction rule (b).

attribute of type string) and a list of Reviewer, which is in turn a record of the three fields RName (string), Rate (integer), and Text (string). Next, NoDoSE then automatically parses them and generates the extraction rules.

SoftMealy: In order to handle missing attributes and attribute permutations in input, Hsu and Dung introduce the idea of finite-state transducer (FST) to allow more variation on extractor structures [10]. A FST consists of two different parts: the *body transducer*, which extract the part of the page that contains the tuples (similar to HLRT in WIEN), and the *tuple transducer* which iteratively extracts the tuples from the body. The tuple transducer accepts a tuple and returns its attributes. Each distinct attribute permutation in the page can be encoded as a successful path from start state to the end state of the tuple transducer; and the state transitions are determined by matching contextual rules that describe the context delimiting two adjacent attributes. Contextual rules consist of individual separators that represent invisible borderlines between adjacent tokens; and an inductive generalization algorithm is used to induce these rules from training examples. Figure 11 shows an example of FST that can be used to extract the attributes of the book reviews: the reviewer name (N), the rating (R), and the comment (T). In addition to the begin and end states, each attribute, A , is followed by a dummy state, $\overline{A}$. Each arc is labeled with the contextual rule that enables the transition and the tokens to output. For example, when the state transition reaches to the R state, the transducer will extract the attribute R until it matches the contextual rules s<R, $\overline{R}$ > (which is composed of s<R, $\overline{R}$ >^{L} and s<R, $\overline{R}$ >^{R}). The state R and the end state are connected if we assume no comment can occur.

STALKER is a WI system that performs hierarchical data extraction [11]. It introduces the concept of embedded catalog (EC) formalism to describe the structure of a wide range of semi-structured documents. The EC description of a page is a tree-like structure in which the leaves are the attributes to be extracted and the internal nodes are lists of tuples. For each node in the tree, the wrapper needs a rule to extract this node from its parent. Additionally, for each list node, the wrapper requires a list iteration rule that decomposes the list into individual tuples. Therefore, STALKER turns the difficult problem of extracting data from an arbitrary complex document into a series of easier extraction tasks from higher level to lower level. Moreover, the extractor uses multi-pass scans to handle missing attributes and multiple permutations. The extraction rules are generated by using of a sequential covering algorithm, which starts from linear landmark automata to cover as many positive examples as possible, and then tries to generate new automata for the remaining examples. A Stalker EC tree that describes the data structure of the running example is shown in Figure 12(a), where some of the extraction rules are shown in Figure 12(b). For example, the reviewer ratings can be extracted by first applying the List(Reviewer) extraction rule (which begins with "
" and ends with "
") to the whole document, and then the Rating extraction rule to each individual reviewer, which is obtained by applying the iteration rule for List(Reviewer). In a way, STALKER is equivalent to multi-pass Softmealy [30]. However, the extraction patterns for each attribute can be sequential as opposed to the continuous patterns used by Softmealy.

DEByE (Data Extraction By Example): Like NoDoSE, DEByE provides an interactive GUI for wrapper generation [24], [25]. The difference is that in DEByE the user marks only atomic (attribute) values to assemble nested tables, while in NoDoSE the user decomposes the whole document in a top-down fashion. In addition, DEByE adopts a bottom-up extraction strategy which is different from other approaches. The main feature of this strategy is that it extracts atomic components first and then assembles them into (nested) objects. The extraction rules, called attribute-value pair patterns (AVPs), for atomic components are identified by context analysis: starting with context length 1, if the number of matches exceeds the estimated number of occurrences provided by the user, it adds additional terms to the pattern until the number of matches is less than the estimated one. For example, DEByE generates AVP patterns, "Name* Reviews", "Name* Rating", "Rating*Text" and "*<li>" for book name, reviewer name, rating and comment respectively (* denotes the data to be extracted). The resulting AVPs are then used to compose an object extraction pattern (OEPs). OEPs are trees containing information on the structure of the document. The sub-trees of an OEP are themselves OEPs, modeling the structure of component objects. At the bottom of the hierarchy lie the AVPs that used to identify atomic components. The assemble of atomic values into lists or tuples is based on the assumption that various occurrences of objects do not overlap each other. For non-homogeneous objects, the user can specify more than one example object, thus creating a distinct OEP for each example.

4.3 Semi-Supervised IE systems

The systems that we categorize as semi-supervised IE systems include IEPAD, OLERA and Thresher. As opposed to supervised approach, OLERA and Thresher accept a rough (instead of a complete and exact) example from users for extraction rule generation, therefore they are called semi-supervised. IEPAD, although requires no labeled training pages, post-effort from the user is required to choose the target pattern and indicate the data to be extracted. All these systems are targeted for record-level extraction tasks. Since no extraction targets are specified for such systems, a GUI is required for users to specify the extraction targets after the learning phase. Thus, users' supervision is involved.

IEPAD is one of the first IE systems that generalize extraction patterns from unlabeled Web pages [31]. This method exploits the fact that if a Web page contains multiple (homogeneous) data records to be extracted, they are often rendered regularly using the same template for good visualization. Thus, repetitive patterns can be discovered if the page is well encoded. Therefore, learning wrappers can be solved by discovering repetitive patterns. IEPAD uses a data structure called PAT trees which is a binary suffix tree to discover repetitive patterns in a Web page. Since such a data structure only records the exact match for suffixes, IEPAD further applies center star algorithm to align multiple strings which start from each occurrence of a repeat and end before the start of next occurrence. Finally, a signature representation is used to denote the template to comprehend all data records. For our running example, only page p_{e2} can be used as input to IEPAD. By encoding each tag as an individual token and any text between two adjacent tags as a special token "T", IEPAD discover the pattern "
TTTT TT
" with two occurrences. The user then has to specify, for example, the 2nd, 4th and 6th "T" tokens, as the relevant data (denoting reviewer name, rating and comment, respectively).

OLERA is a semi-supervised IE system that acquires a rough example from the user for extraction rule generation [32]. OLERA can learn extraction rules for pages containing single data records, a situation where IEPAD fails. OLERA consists of 3 main operations. (1) *Enclosing an information block of interest*: where the user marks an information block containing a record to be extracted for OLERA to discover other similar blocks (using approximate *matching* technique) and generalize them to an extraction pattern (using *multiple string alignment* technique). (2) *Drilling-down/rolling-up an information slot*: drilling-down allows the user to navigate from a text fragment to more detailed components, whereas rolling-up combines several slots to form a meaningful information unit. (3) *Designating relevant information slots* for schema specification as in IEPAD.

Thresher [33] is also a semi-supervised approach that is similar to OLERA. The GUI for Thresher is built in the Haystack browser which allows users to specify examples of semantic contents by highlighting them and describing their meaning (labeling them). However, it uses tree edit distance (instead of string edit distance as in OLERA) between the DOM subtrees of these examples to create a wrapper. Then it allows the user to bind the semantic web language RDF (Resource Description Framework) classes and predicates to the nodes of these wrappers.

4.4 Un-Supervised IE systems

As shown at the left-top of Figure 5, unsupervised IE systems do not use any labeled training examples and have no user interactions to generate a wrapper. Unsupervised IE systems, RoadRunner and EXALG, are designed to solve page-level extraction task, while DeLa and DEPTA are designed for record-level extraction task. In contrast to supervised IE systems where the extraction targets are specified by the users, the extraction target is defined as the data that is used to generate the page or non-tag texts in data-rich regions of the input page. In some cases, several schemas may comply with the training pages due to the presence of nullable data attributes, leading to ambiguity [34]. The choice of determining the right schema is left to users. Similarly, if not all data is needed, post-processing may be required for the user to select relevant data and give each piece of data a proper name.

DeLa: As an extension of IEPAD, DeLa [35], [36] removes the interaction of users in extraction rule generalization and deals with nested object extraction. The wrapper generation process in DeLa works on two consecutive steps. First, a *Data-rich Section Extraction* algorithm (DSE) is designed to extract data-rich sections from the Web pages by comparing the DOM trees for two Web pages (from the same Web site), and discarding nodes with identical sub-trees. Second, a *pattern extractor* is used to discover continuously repeated (C-repeated) patterns using suffix trees. By retaining the last occurrence for each discovered pattern, it discover new repeated patterns from the new sequence iteratively, forming nested structure. For example, given the string sequence "<P><A>T</A><A>T </A>T</P><P><A>T</A>T</P>", DeLa will discover "<P><A>T</A>T<P>" from the immediate sequence "<P><A>T</A>T</P><P><A>T</A>T</P>" and return parenthesized pattern "(<P>(<A>T</A>)*T<P>)*" to denote the nested structure. Since a discovered pattern may cross the boundary of a data object, DeLa tries K pages and selects the one with the largest page support. Again, each occurrence of the regular expression represents one data object. The data objects are then transformed to a relational table where multiple values of one attribute are distributed into multiple rows of the table. Finally, labels are assigned to the columns of the data table by four heuristics, including element labels in the search form or tables of the page and maximal-prefix and maximal-suffix shared by all cells of the column.

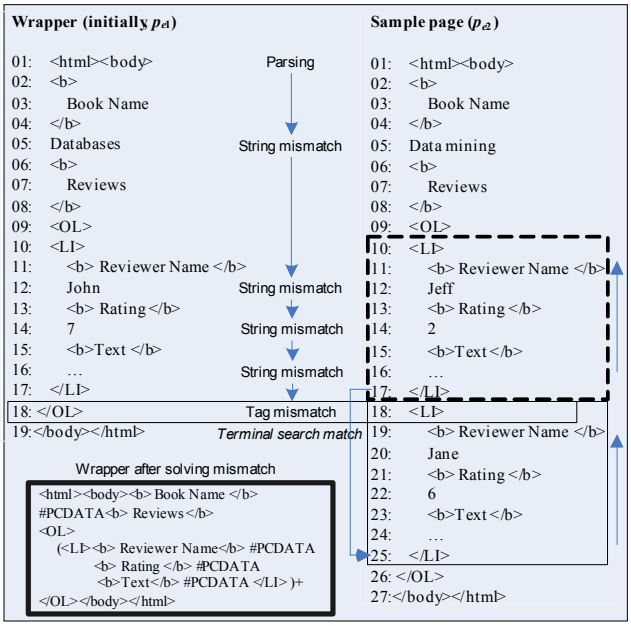

Fig. 13. Matching the first two pages of the running example (taken from [22]).

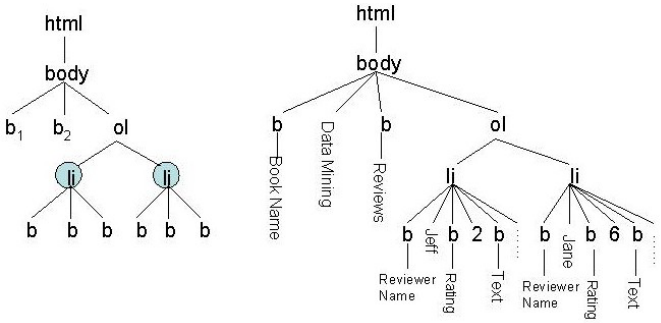

Fig. 14. The tag tree (left) and the DOM tree (as a comparison) for page p_{e2} in Figure 4.

RoadRunner considers the site generation process as encoding of the original database content into strings of HTML code [22]. As a consequence, data extraction is considered as a decoding process. Therefore, generating a wrapper for a set of HTML pages corresponds to inferring a grammar for the HTML code. The system uses the *ACME matching* technique to compare HTML pages of the same class and generate a wrapper based on their similarities and differences. It starts from comparing two pages, using the ACME technique to align the matched tokens and collapse for mismatched tokens. There are two kinds of mismatches: *string mismatches* that are used to discover attributes (#PCDATA) and *tag mismatches* that are used to discover iterators (+) and optional (?). Figure 13 shows both an example of matching for the first two pages of the running example and its generated wrapper. Since there can be several alignments, RoadRunner adopts UFRE (union-free regular expression) to reduce the complexity. The alignment result of the first two pages is then compared to the third page in the page class. In addition to the module for template deduction, RoadRunner also includes two modules, Classifier and Labeler to facilitate wrapper construction. The first module, *Classifier*, analyzes pages and collects them into clusters with a homogeneous structure, i.e. pages with the same template are clustered together. The second module, *Labeler*, discovers attribute names for each page class.

EXALG: Arasu and Molina presented an effective formulation for the problem of data extraction from Web pages [37]. The input of EXALG is a set of pages created from the unknown template T and the values to be extracted. EXALG deduces the template T and uses it to extract the set of values from the encoded pages as an output. EXALG detects the unknown template by using the two techniques *differentiating roles* and *equivalence classes* (EC). In the former technique, the occurrences with two different paths of a particular token have different roles. For example, in the running example, the role of "Name" when it occurs in "Book Name" (i.e., $Name_5$) is different from its role when it occurs in "Reviewer Name" (i.e., $Name_{14}$). In the later technique, an equivalence class is a maximal set of tokens having the same occurrence frequencies over the training pages (*occurrence-vector*). For example, in Figure 4, the two tokens $<html>_1$ and $<body>_2$ have the same occurrence-vector (<1, 1, 1, 1>), so they belong to the same equivalence class. The insight is that template tokens that encompass a data tuple have the same occurrence vector and form an equivalence class. However, to avoid data tokens to accidentally form an equivalence class, ECs with insufficient support (the number of pages containing the tokens) and size (the number of tokens in an EC) are filtered. In addition, to conform to the hierarchical structure of the data schema, equivalence classes must be mutually nested and the tokens in an EC must be ordered. Those valid ECs are then used to construct the original template.

DEPTA (Data Extraction based on Partial Tree Alignment): Like IEPAD and DeLa, DEPTA can be only applicable to Web pages that contain two or more data records in a data region. However, instead of discovering repeat substring based on suffix trees, which compares all suffixes of the HTML tag strings (as the encoded token string described in IEPAD), it compares only adjacent substrings with starting tags having the same parent in the HTML tag tree (similar to HTML DOM tree but only tags are considered). The insight is that data records of the same data region are reflected in the tag tree of a Web page under the same parent node. Thus, irrelevant substrings do not need to be compared together as that in suffix-based approaches. Furthermore, the substring comparison can be computed by string edit distance instead of exact string match when using suffix trees where only completely similar substrings are identified. The described algorithm, called MDR [38], works in three steps. First, it builds an HTML tag tree for the Web page as shown in Figure 14 where text strings are disregarded. Second, it compares substrings for all children under the same parent. For example, we need to make two string comparison, (b_1, b_2) and (b_2, ol), under parent node <body>, where the tag string node <ol> is represented by

"
<b><b><b>
<b><b><b>". If the similarity is greater than a predefined threshold (as shown in the shaded nodes in Figure 14), the nodes are recorded as data regions. The third step is designed to handle situations when a data record is not rendered contiguously as assumed in previous works. Finally, the recognition of data items or attributes in a record is accomplished by partial tree alignment [39]. Tree alignment is better than string alignment for it considers tree structure, thus, reducing the number of possible alignments. The algorithm first chooses the record tree with the largest number of data items as center and then matches other record trees to the center tree. However, DEPTA only adds tag nodes to the center tree when the positions of the tag nodes can be uniquely determined in the center tree. For remained nodes, they are processed in the next iteration after all tag trees are processed. Note that DEPTA assumes that non-tag tokens are data items to be extracted, thus, it extracts not only the reviewer name, rating and comments, but also the labels "Reviewer Name", "Rating", and "Text" for page p_{e2} in our running example. Further, DEPTA is limited to handle nested data records. So, a new algorithm, NET, is developed to handle such data records by performing a post-order traversal of the visual-based tag tree of a Web page and matching subtrees in the process using a tree edit distance method and visual cues [40].

Of the unsupervised WI approaches, one important issue is to differentiate the role of each token: either a data token or template token. Some assume that every HTML tag is generated by the template and other tokens are data items to simplify the issue (as in DeLa and DEPTA). However, the assumption does not hold for many collections of pages (therefore, IEPAD and OLERA simply leave the issue to distinguish between data and template tokens to the users). RoadRunner also assumes that every HTML tag is generated by the template, but other matched string tokens are also considered as part of the template. In comparison, EXALG has the most detailed tokenization method while more flexible assumption where each token can be a template token if there are enough tokens to form frequently occurring equivalence class.

On the other hand, DEPTA conducts the mining process from single Web pages, while RoadRunner and EXALG do the analysis from multiple Web pages (While DeLa takes advantages of multiple input pages for data-rich section extraction and generalized pattern construction, it discovers C-repeat patterns from single Web pages.). The later, in our viewpoint, is the key point that is used to differentiate the role of each token. Thus, multiple pages of the same class is also used to discover data rich section (as in DeLa) or eliminate noisy information (as in [41]). Meanwhile, the adaptation of tree matching in DEPTA (as well as Thresher) also provides better result than string matching techniques used in IEPAD and RoadRunner. EXALG similarly does not make full use of the tree structure although the DOM tree path information is used for differentiating token roles. Finally, since information extraction is only a part of a wrapper program or information integration systems, additional tasks like page fetching, label assignment, and mapping with other web data sources are remained to be processed.

Due to space limitation, we are not able to compare all researches here. For example, ViNTs [42] is a record-level wrapper generation system which exploits visual information to find separators between data regions from search result pages. However, the algorithm can be only applicable to pages that contain at least four data records. Another related approach that has been applied on Web sites for extracting information from tables is [43]. The technique relies on the use of additional links to a detail page containing additional information about that item. In parallel to the efforts to detect Web tables, other researchers have worked in detecting tables in plain text documents (such as government statistical reports) and segmenting them into records [44]. Since these approaches do not address the problem of distinguish data tokens from template tokens, we consider them as semi-supervised approaches.

5 A Comparative Analysis of IE Tools

Although many researchers have developed various tools for data extraction from Web pages, there has been only a limited amount of effort to compare such tools. Unfortunately, in only a few cases can results generated by distinct tools be directly comparable. From our viewpoint, even in these few cases, the main goal of the comparison is for a survey. Therefore, in this section, we use the criteria of the 3 dimensions suggested in section 3 to compare the surveyed IE tools.

5.1 Task Domain-based comparison

In this section, we contrast among the capabilities of the surveyed IE systems to support various IE tasks as shown in Table 1. The features in this dimension include input variation, such as page type, Non-HTML support, and output variation such as extraction level, attribute variation and template variation.

Page Type: We first compare the input documents that each IE system targets. As discussed above, Web pages may be *structured*, *semi-structured* or *free-text* Web pages according to the level of structurization. For example, manual or supervised IE systems are designed to extract information from cross-website pages (e.g. professor data from various universities), while semi-supervised and supervised IE systems are designed primarily for extracting data from the deep Web (template pages). Thus, the latter systems depend heavily on the common template that is used to generate Web pages, while the former have included more features of the tokens (e.g. the number of characters, the fraction of upper-case letters, etc.) for inducing extraction rules. By incorporating more characteristics of the template pages, unsupervised IE systems present high-degree automation for extraction rule generalization; in contrast, the extension to non-template pages is rather limited.

TABLE 1
ANALYSIS BASED ON THE TASK DOMAINS

Tools		Page Type	NHS	Extraction Level	Extraction Targets Variation			Template Variation		UTA
					MA/MVA	MOA	Nested	VF	CT	
Manual	Minerva	Semi-S	Yes	Record Level	Yes	Yes	Yes	Both	By Order	Yes
	TSIMMIS	Semi-S	Yes	Record Level	Yes	No	Yes	Disj	By Order	No
	WebOQL	Semi-S	No	Record Level	Yes	Yes	Yes	Disj	By Order	No
	W4F	Temp	No	Record Level	Yes	Yes	Yes	SP	By Order	Yes
	XWRAP	Temp	No	Record Level	Yes	No	Yes	SP	By Order	Yes
Supervised	RAPIER	Free	Yes	Field Level	Yes	--	--	Disj	More constraints	Yes
	SRV	Free	Yes	Field Level	Yes	--	--	Disj	More constraint	Yes
	WHISK	Free	Yes	Record Level	Yes	Yes	No	Disj	By Order	Yes
	NoDoSE	Semi-S	Yes	Page/Record	Yes	Limited	Yes	No	By Order	No
	DEByE	Semi-S	Yes	Record Level	Yes	Yes	Yes	Disj	More constraint	No
	WIEN	Semi-S	Yes	Record Level	No	No	Limited	No	By Order	No
	STALKER	Semi-S	Yes	Record Level	Yes	Yes	Yes	Both	More constraint	No
	SoftMealy	Semi-S	Yes	Record Level	Yes	Limited	Multi Pass	Disj	By Order/ SinglePass	Yes
Semi-Supervised	IEPAD	Temp	Limited	Record Level	Yes	Limited	Limited	Both	By Order	Yes
	OLERA	Temp	Limited	Record Level	Yes	Limited	Limited	Both	By Order	Yes
Un-Supervised	DeLa	Temp	Limited	Record Level	Yes	Limited	Yes	Both	By Order	No
	RoadRunner	Temp	Limited	Page Level	Yes	No	Yes	No	By Order	No
	EXALG	Temp	Limited	Page Level	Yes	No	Yes	Both	By Order	No
	DEPTA	Temp	No	Record Level	Yes	No	Limited	Disj	By Order	No

Non-HTML Support (NHS): The support for non-HTML inputs depends on the features or background knowledge used by the IE systems. Thus, when an IE system fails to generalize extraction rules for an IE task, we (the programmers) know how to or what to adjust the system for such a task. Most supervised systems can support non-HTML documents by modifying the generalization hierarchy (e.g. Softmealy) or adding new token features (e.g. SRV). Manual systems such as Minerva and TSIMMIS, where extraction rules are written by hand, can be adapted by the wrapper developer to handle non-HTML documents. Some wrappers, e.g. WebOQL, W4F, XWrap, and DEPTA, rely heavily on the use of DOM trees information in their systems, so they cannot support non-HTML documents, while sequence based approaches, such as IEPAD, OLERA, RoadRunner, and DeLa can be adapted to handle non-HTML documents by adding proper encoding schemes. The equivalence class technology of EXALG also supports non-HTML documents, but the success depends on token role differentiation.

Extraction Level: IE tasks can be classified into four categories: field-level, record-level, page-level and site-level. Rapier and SRV are designed to extract single-slot records, or equivalently field-level extractions. Wrappers in EXALG and RoadRunner extract the embedded data objects in whole pages which may contain records of multiple kindes, so wrappers in these systems are page-level. The other remaining systems in Table 1 are examples of record-level IE tasks, although some can be extended for page-level extraction, e.g. NoDoSE, STALKER, etc. Most record-level IE systems discover record boundaries and then divide them into separate items, while the bottom-up extraction strategy in DEByE extracts a set of attributes and then assembles them to form a record. So far, there are no site-level IE systems.

Extraction Target Variation: Many Web pages are hierarchically organized with multiple nesting levels. Typically, this complex structure is loose, presenting variations on semi-structured data. The complex degree of an extraction target (data object) depends on the appearance of missing attributes (MA), multiple-valued attributes (MVA), multi-

ordering attributes (MOA), and nested data objects. To handle these variations, the extracting procedure needs special care in addition to its usual logic where attributes appear exactly once without ordering and nesting issues. Understanding how various IE systems support these variations can help us decide how to tailor an IE system to new tasks. Note that for field-level extraction systems (SRV and Rapier), handling of these variations does not present specific difficulties, since they do not deal with the relationships of attributes in the data objects.

Most IE systems support missing attributes and multiple-valued attributes extraction, except for WIEN and WHISK. The special care for programming-based IE systems is usually an exception handler, e.g. Minerva, W4F and WebOQL. In TSIMMIS, two operators "case" and "split" are designed to handle missing attributes and multiple-valued attributes. Many IE systems do not support multiple-ordering attributes since their extraction rules depend on the location of the fields within a record. Hsu was a pioneer who attempted to overcome the problem of multiple ordering attributes. However, from our viewpoint, the situations he handled were instances of missing attributes. So, we consider that SoftMealy is limited to handle MOA using single-pass finite state transducer (FST). The use of FST in SoftMealy, also make it possible to handle MA and MVA. In overall, SoftMealy can handle objects of nested structures through multi-pass FST. Stalker can handle MOA and nested object extraction by multi-pass scans over the input data. Other IE systems (IEPAD, OLERA and DeLa), make use of alignment technique to form disjunctive rules to handle MA, MVA, MOA. In addition, the use of multiple encoding schemes in IEPAD and OLERA give them the opportunity to handle more complex nested data objects. The two heuristic-based mining components in NoDoSE and the bottom-up strategy (where the set of attributes are recognized, extracted and stored in a set variable prior to the object itself) in DEByE give these systems the ability to handle MOA and nested data objects in overall. RoadRunner and EXALG did not support MOA because their extraction rules depend on the location of the attributes within a record, although in overall, they can handle nested data objects. DEPTA, theoretically can support nested data objects by exploiting the tag tree structure. MOA is not possible in DEPTA since the partial tree match is based on unique order of the tag children with the same parent.

Template Variation: The difficulties in extraction rule induction come from the variant formats of the data instances. As described in Section 3.1, an attribute may have variant formats (VF), which usually require disjunctive rule supports or sequential rule supports. Some IE systems support both disjunctive rules and sequential patterns (SP) for rule generalization. To the best of our knowledge, WIEN, W4F, XWrap, NoDoSE, and RoadRunner do not support disjunctive rules. However, W4F and XWrap support sequential pattern for rule generalization. A regular expressions containing don't care symbols is an example of sequential pattern. Sequential patterns can be generalized by alignment technique or by sequential pattern mining (e.g. Stalker). Meanwhile, different attributes may have the same display format called common format (CT). Most IE systems take the advantage of attribute order to extract them. Others, e.g. DeBYE and Stalker, add more constraints to form a longer extraction rule. What follows is that the extraction precision can be greatly decreased in case of missing attributes or multiple-order attributes.

UnTokenized Attributes (UTA): So far, we've seen three approaches to handle untokenized attributes. The first one is through post-processing. For example, the split operator in W4F offers regular expressions and constraints to address finer pieces of data. The second one is by contextual rules instead of delimiter-based rules. As proposed by Softmealy, the idea of separators as well as contextual rules helps user address data of any granularity. Finally, multiple-level encodings also allow IE systems to address data of different granularity without sacrificing the advantage of abstraction for rule generalization as in IEPAD and OLERA.

5.2 Technique-based comparison

In this section, we use the criteria suggested in Section 3.2 to compare and evaluate IE systems from the perspective of the underlying techniques used. The results are shown in Table 2 and discussed below.

Scan Pass: This comparison refers to the number of scan passes required over an input document for information extraction. Most WI systems design the extractor to scan the input document once, referred to as single-pass extractor, while others (e.g. STALKER and multi-pass SoftMealy) scan the input document several times to complete the extraction. The extractor of DEByE also needs multiple passes to extract each atomic attributes. Generally speaking, single-pass wrappers are more efficient than multi-pass wrappers. However, multi-pass wrappers are more effective at handling data objects with unrestricted attribute permutations or complex object extraction. SRV and Rapier can only generate single slot rules, so the extractor needs to make multiple passes over the input page to extract relevant data.

Extraction Rule Type: Most WI systems use extraction rules that are represented as regular grammars to identify the beginning and end of the relevant data, whereas Rapier and SRV use extraction rules expressed using first order logic. Regular expression rules are powerful for semi-structured inputs, especially template-based pages, since we usually find common tokens surrounding the data to be extracted. Even when no common tokens exist, we can induce rules by incorporating a generalization hierarchy of tokens as background knowledge (e.g. Softmealy). However, for free-text inputs, where very few common tokens can be found, we need to incorporate more features, e.g. digit density, length, POS tags, etc. to generalize the common characteristics among various tokens. That's why first-order logic rules are used for free-text IE tasks (e.g. SRV and Rapier).

TABLE 2
ANALYSIS BASED ON THE TECHNIQUES USED

Tools	Scan Pass	Extraction Rule Type	Features Used	Learning Algorithm	Tokenization Schemes
Minerva	Single	Regular exp.	HTML tags/Literal words	None	Manually
TSIMMIS	Single	Regular exp.	HTML tags/Literal words	None	Manually
WebOQL	Single	Regular exp.	Hypertree	None	Manually
W4F	Single	Regular exp.	DOM tree path addressing	None	Tag Level
XWRAP	Single	Context-Free	DOM tree	None	Tag Level
RAPIER	Multiple	Logic rules	Syntactic/Semantic	ILP (bottom-up)	Word Level
SRV	Multiple	Logic rules	Syntactic/Semantic	ILP (top-down)	Word Level
WHISK	Single	Regular exp.	Syntactic/Semantic	Set covering (top-down)	Word Level
NoDoSE	Single	Regular exp.	HTML tags/Literal words	Data Modeling	Word Level
DEByE	Multiple	Regular exp.	HTML tags/Literal words	Data Modeling	Word Level
WIEN	Single	Regular exp.	HTML tags/Literal words	Ad-hoc (bottom-up)	Word Level
STALKER	Multiple	Regular exp.	HTML tags/Literal words	Ad-hoc (bottom-up)	Word Level
SoftMealy	Both	Regular exp.	HTML tags/Literal words	Ad-hoc (bottom-up)	Word Level
IEPAD	Single	Regular exp.	HTML tags	Pattern Mining, String Alignment	Multi-Level
OLERA	Single	Regular exp.	HTML tags	String Alignment	Multi-Level
DeLa	Single	Regular exp.	HTML tags	Pattern Mining	Tag Level
RoadRunner	Single	Regular exp.	HTML tags	String Alignment	Tag Level
EXALG	Single	Regular exp.	HTML tags/Literal words	Equivalent Class and Role Differentiation by DOM tree path	Word Level
DEPTA	Single	Tag Tree	HTML tags treeHTML tags	Pattern Mining, String comparison, Partial tree alignment	Tag Level

Features Used: Earlier IE systems are designed to handle non-template based Web pages, say computer science department Web pages from various universities. Therefore, they have used both HTML tags and literal words as delimiter-based constraints. For template-based Web pages, it is possible to use DOM tree paths to denote a specific piece of information in a Web page. For example, W4F, XWrap and other commercial products use DOM tree paths to address a Web page. Since the data to be extracted are often co-located in the same path of the DOM tree, this makes the rule learning process much easier. For free text information extraction, natural language processing techniques such as part-of-speech tagger and Word-Net semantic classes are used as additional features. SRV also uses orthographic features, token's length, and link grammars. Finally, EXALG exploits statistical information of the tokens in Web pages to generate their wrappers.

Learning Algorithm: Wrappers in programming-based WI systems are written by hand and take as input a specification that is declaratively stated where the data of interest is located in the HTML pages and how the data is packaged into objects. Thus, no learning algorithms are used in these systems. Rapier is a bottom-up relational learning system inspired by ILP methods, while SRV is a top-down relational algorithm. Whisk is a top-down covering learning system. Its patterns have two components that specify the *context* and the *exact delimiters* of the phrase to be extracted. DEByE and NoDoSE all require a large amount of support from users to model the data in the documents. They focus on the interface design and apply very simple methods to learn extraction patterns, i.e. common prefix and suffix of the data values to be extracted. On the other hand, Stalker and SoftMealy use Ad-hoc generalization methods for learning extraction rules. They focus on the learning techniques and the extractor architecture and use a hierarchy of token classes for token generalization, which is quite different from NoDoSE and DEByE where the extraction rules are simply based on superficial or literal words.

Semi-supervised or unsupervised IE systems mainly apply data mining techniques for various pattern discoveries. IEPAD discovers regular and adjacent maximum patterns using PAT trees and string alignment techniques, while DeLa further discovers nested structures from continuous repeated (C-repeated) patterns. OLERA applies approximate string matching and string alignment techniques following the users' enclosing, drill-down/roll-up operations. RoadRunner analyzes input pages by string comparison using the ACME technique. EXALG exploits statistical information to generate the template and schema of Web pages by using *equivalence classes* and *differentiating roles* techniques. DEPTA applies a mining technique and partial tree alignment to mine data records in a Web page. In comparison, IEPAD and DEPTA discover repeated patterns

TABLE 3
ANALYSIS BASED ON AUTOMATION DEGREE

Tools	User Expertise	Fetch support	Output/API Support	Applicability	Limitation
Minerva	Programming	No	XML	High	Not restricted
TSIMMIS	Programming	No	Text	High	Not restricted
WebOQL	Programming	No	Text	High	Not restricted
W4F	Programming	Yes	XML	Medium	Not restricted
XWRAP	Programming	Yes	XML	Medium	Not restricted
RAPIER	Labeling	No	Text	Medium	Not restricted
SRV	Labeling	No	Text	Medium	Not restricted
WHISK	Labeling	No	Text	Medium	Not restricted
NoDoSE	Labeling	No	XML, OEM	Medium	Not restricted
DEByE	Labeling	Yes	XML, SQL DB	Medium	Not restricted
WIEN	Labeling	No	Text	Medium	Not restricted
STALKER	Labeling	No	Text	Medium	Not restricted
SoftMealy	Labeling	Yes	XML, SQL DB	Medium	Not restricted
IEPAD	Post labeling Pattern selection	No	Text	Low	Multiple-records page
OLERA	Partial Labeling	No	XML	Low	Not restricted
DeLa	Pattern selection	Yes	Text	Low	Multiple-records page, More than one page
RoadRunner	Pattern selection	Yes	XML	Low	More than one page
EXALG	Pattern selection	No	Text	Low	More than one page
DEPTA	Pattern selection	No	SQL DB	Low	Multiple-records pages

from one HTML page, while Roadrunner and EXALG discover repeat patterns from multiple HTML pages.

Tokenization Schemes: Wrappers in Minerva and TSIMMIS are written by hand, so they do not need to tokenize the input pages. Most WI systems for Web pages support tag-level tokernization. Some systems even support word-level tokernization, e.g. supervised WI systems and EXALG. WebOQL, W4F, XWrap, RoadRunner and DeLa use a tag-level encoding scheme to translate the input training pages into tokens. Also, the input HTML page in W4F and XWrap has been parsed to construct a parse tree that reflects its HTML tags hierarchy following the document object model (DOM). Finally, IEPAD and OLERA allow multiple levels of encodings for input training pages.

5.3 Automation degree-based comparison

In this section, we use the features suggested in Section 3.3 to compare and evaluate IE systems from the automation degree prospective. The results are shown in Table 3 and discussed below.

User Expertise: Manual IE systems require users of programming background to write correct extraction rules. Supervised and semi-supervised WI systems require users to label exact or part of the data to be extracted, thus there is no special expertise needed. For unsupervised systems, they require no assistant from users (except for pattern selection). For IEPAD and OLERA, although they require no labeling before pattern discovery, post-labeling is needed to sift desired data, while the work of distinguishing template tokens from data tokens is accomplished by unsupervised IE systems. Strictly speaking, the label of the data extracted by unsupervised IE systems remains to be assigned, and only DeLa has dealt with this problem.

Fetching Support: Most IE systems focus on extraction rule generalization and use a set of pages that are manually downloaded as training examples. Some systems specifically support page fetching in wrapper construction. For example, W4F has a component called *RetrieveAgent* that is used to retrieve a Web source by inputting its URL. Also, the *syntactical normalizer* component of XWrap accepts an URL entered by the user, issues an HTTP request to the remote server identified by the URL and fetches the corresponding Web page. Other systems also propose new tools for page fetching support. For instance, WNDL is a language proposed by Hsu et al. to describe Web navigation for page fetching support with Softmealy and IEPAD [45]. ASByE, a member of DEByE family, is a tool for collecting static and dynamic Web pages. DeLa uses the existing Hidden Web crawler, *HiWe*, to automatically collect the labels of the elements from Web sites and send queries to the Web site.

Output/API Support: Outputting the extracted relevant

data is comparably simple, so most IE systems support it. The systems Minerva, W4F, XWrap, NoDoSE, DEByE, SoftMealy, OLERA and RoadRunner output the extracted data in a XML format. Also, NoDoSE supports other formats, such as OEM, and DEByE supports SQL database output format. On the other hand, API support is important since it is the connection between the generated wrapper and information integration systems. Programming-based IE systems have API supports, while others do not specifically mention this in their papers.

Applicability: As described in section 3.3, applicability concerns how easy these approaches can be extended to other task domains. A key factor for high applicability is that domain-specific information is separate from the underlying learning mechanism. For the various IE tasks we discussed above, manual systems and supervised systems have good modularity while semi-supervised or unsupervised systems have less applicability since they have pushed the domain specific information to the limit for high automation degree.

Limitation: Finally, we consider the requirements for multiple data-records or multiple training pages input. Although, we can regard such requirements as different input IE task, we view them as a limitation of these approaches for various WI systems to be compared in the same task domain. Take template-page IE for example, an IE system that needs multiple-records training Web pages can not be applied to a site that includes Web pages of a single record. As summarized in Table 3, there is no restriction about the content and the number of training pages for manual and supervised IE systems. IEPAD, DeLa and DEPTA require input pages with multiple-records to generate a wrapper. DeLa, RoadRunner, EXALG require more than one training page as input for their approaches to work.

5.4 Overall comparison

Although we have compared various IE systems from three dimensions, there are correlations among these criteria. For example, template-based pages have higher automation degree than non-template pages and free-text documents since the inputs present structured framework that can be discovered by unsupervised approaches. However, this does not imply that data extraction from template-based pages is easier than other pages. Instead, new problems arise, e.g. distinction between template and data tokens, and label assignment to data tokens.

As shown in Figure 15, manual IE systems can be applied to all kinds of inputs as long as proper features are provided by the systems, though it depends on the programmers' techniques to compose the extraction rules. Semi-supervised and unsupervised IE systems can be applied only to template-based pages since their success rely on the existence of template. In addition, we also see that unsupervised systems usually apply superficial features such as HTML tags for regular expression rules since they are targeted for template-based pages. For IE from cross-site pages and free texts, semantic features (e.g. orthographic features, token's length, etc.) are required since there are less common tags and words among the input documents.

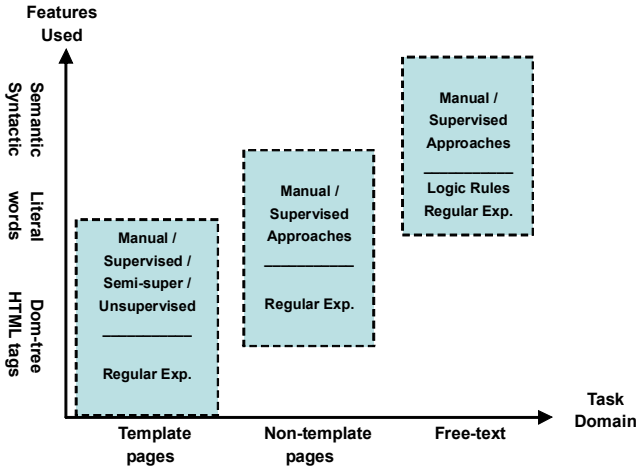

Fig. 15. Overall comparison.

For a practitioner, one wants to know which techniques are effective, good recall and precision. Since these systems deal with different data and have different features, it is not possible to evaluate them in a consistent way. Thus, we can only compare them from their applicability. Semi-supervised and unsupervised IE systems have embedded in their systems heuristics observed from template pages, e.g. contiguous data area (IEPAD), non-contiguous data records (DEPTA), nested data objects (DeLa). Since there are many variations on the Web, there is no guarantee such techniques work for all Web pages, though we do find that newly proposed approaches can solve more pages than past approaches. As for supervised approaches, since data to be extracted are labeled by users, their applicability is comparatively better than unsupervised systems. Still, there is no guarantee for the success of rule induction.

For a researcher, one wants to know which technique to apply when tailoring current systems to a new IE task domain. As discussed above, the techniques used in unsupervised IE systems is hard to extend to free texts and even non-template pages since many heuristics are applicable only to template-based pages. For supervised approaches, we have seen well-known learning techniques (e.g. ILP and set covering in SRV, WHISK, etc.) as well as Ad-hoc learning (bottom-up generalization in Stalker, Softmealy, etc.). Ad-hoc learning techniques are faster in learning by incorporating a token hierarchy for generalization. We appreciate supervised approaches since we can add new features to existing systems without modifying the learning algorithms. Although only ILP and set covering algorithms are used now, it would be interesting to see other learning algorithms (e.g. support vector machine, etc.) to be applied.

6 Conclusions and Future Work

In this paper, we survey the major IE tools in the literature and compare them in three dimensions: the task domain, the automation degree, and the techniques used. A set of criteria are proposed for the comparison and evaluation in each dimension. The criteria of the first dimension explain why an IE system fails to handle some Web sites of particu-

lar structures. The criteria of the second dimension measure the degree of automation for IE systems. The criteria of the third dimension measure the performance of IE systems. We present our taxonomy of WI systems from the users' viewpoint and compare important features of WI systems that affect their effectiveness.

There are several points to make from the survey. First, we see the trend of developing highly automatic IE systems, which saves not only the effort for programming, but also the effort for labeling. Thus, although the creation of Web services provides another way for data exchange and information integration, it may not be the best choice since the involvement of programmer is unavoidable. On the other hand, not all IE tasks can be wrapped by fully automatic IE systems. Unsupervised approaches can only support template pages. The extension of such systems to non-template page extraction tasks is very limited. In contrast, supervised approaches, although require annotations from users, extend well to non-template page extraction if proper features are selected for extraction rules.

The technique of information extraction can be applied to non-HTML documents such as medical records and curriculum vitae to facilitate the maintenance of large semi-structured documents. In the future, information extraction from cross-website pages will become more important as we move toward semantic Web. In this survey, we only focus on data extraction from Web documents. Page fetching support and extracted data integration (or schema mapping) from various data sources are two research topics that are not thoroughly studied in this paper. A new research topic on integration of search forms has also drawn many attentions [46], [47].

REFERENCES

This work was partially sponsored by National Science Council, Taiwan under grant NSC94-2213-E-008-020 and NSC94-2524-S-008-002.

REFERENCES

[1] Riloff, E., Automatically constructing a dictionary for information extraction tasks. Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), pp. 811-816, AAAI Press/The MIT Press, 1993.

[2] Huffman, S., Learning information extraction patterns from examples. Connectionist, statistical, and symbolic Approaches to Learning for Natural Language Processing, Springer-Verlag, 1996.

[3] Kim, J. and Moldovan, D., Acquisition of linguistic patterns for knowledge-based information extraction. IEEE Transactions on Knowledge and Data Engineering 7(5): 713-724, 1995.

[4] Krupka, G., Description of the SRA system as used for MUC-6. Proceedings of the sixth Message Understanding Conference (MUC-6), pp. 221-235, 1995.

[5] Soderland, S., Fisher, D., Aseltine, J., and Lehnert, W., CRYSTAL: Inducing a conceptual dictionary. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[6] Soderland, S., Learning information extraction rules for semi-structured and free text. Journal of Machine Learning, 34(1-3): 233-272, 1999.

[7] Califf, M. and Mooney, R., Relational learning of pattern-match rules for information extraction. Proceedings of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing Stanford, California, March, 1998.

[8] Freitag, D., Information extraction from HTML: Application of a general learning approach. Proceedings of the Fifteenth Conference on Artificial Intelligence (AAAI-98).

[9] Kushmerick, N., Weld, D., and Doorenbos, R., Wrapper induction for information extraction. Proceedings of the Fifteenth International Conference on Artificial Intelligence (IJCAI), pp. 729-735, 1997.

[10] Hsu, C.-N. and Dung, M., Generating finite-state transducers for semi-structured data extraction from the web. Journal of Information Systems 23(8): 521-538, 1998.

[11] Muslea, I., Minton, S., and Knoblock, C., A hierarchical approach to wrapper induction. Proceedings of the Third International Conference on Autonomous Agents (AA-99), 1999.

[12] Chang, C.-H., Hsu, C.-N., and Lui, S.-C. Automatic information extraction from semi-Structured Web Pages by pattern discovery. Decision Support Systems Journal, 35(1): 129-147, 2003.

[13] Kushmerick. N., Adaptive Information Extraction: Core technologies for Information agents. In Intelligent Information Agents R&D in Europe: An AgentLink perspective (Klusch, Bergamaschi, Edwards & Petta, eds.). Lecture Notes in Computer Science 2586, Springer, 2003.

[14] Soderland, S., Learning to extract text-based information from the world wide web. Proceedings of the third International Conference on Knowledge Discovery and Data Mining (KDD), pp. 251-254, 1997.

[15] Ciravegna, F., Learning to tag for information extraction from text. Proceedings of the ECAI-2000 Workshop on Machine Learning for Information Extraction, Berlin, August 2000.

[16] Laender, A. H. F., Ribeiro-Neto, B., DA Silva and Teixeira, A brief survey of Web data extraction tools. SIGMOD Record 31(2): 84-93, 2002.

[17] Crescenzi, V., and Mecca, G., Grammars have exceptions. Information Systems, 23(8): 539-565, 1998.

[18] Hammer, J., McHugh, J. and Garcia-Molina, Semistructured data: the TSIMMIS experience. In Proceedings of the 1st East-European Symposium on Advances in Databases and Information Systems (ADBIS), St. Petersburg, Rusia, pp. 1-8, 1997.

[19] Arocena, G. O. and Mendelzon, A. O., WebOQL: Restructuring documents, databases, and Webs. Proceedings of the 14th IEEE International Conference on Data Engineering (ICDE), Orlando, Florida, pp. 24-33, 1998.

[20] Saiiuguet, A. and Azavant, F., Building intelligent Web applications using lightweight wrappers. Data and Knowledge Engineering 36(3): 283-316, 2001.

[21] Liu, L., Pu, C., and Han, W. XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources, Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE), San Diego, California, pp. 611-621, 2000.

[22] Crescenzi, V., Mecca, G. and Merialdo, P., RoadRunner: towards-automatic data extraction from large Web sites. Proceedings of the 26th International Conference on Very Large Database Systems (VLDB), Rome, Italy, pp. 109-118, 2001.

[23] Adelberg, B., NoDoSE: A tool for semi-automatically extracting structured and semi-structured data from text documents. SIGMOD Record 27(2): 283-294, 1998.

[24] Laender, A. H. F., Ribeiro-Neto, B. and DA Silva, A., S., DEByE -

Data Extraction by Example. Data and Knowledge Engineering, 40(2): 121-154, 2002.

[25] Ribeiro-Neto, B., A., Laender, A., H., F. and DA Silva, A., S., Extracting semi-structured data through examples. Proceedings of the Eighth ACM International Conference on Information and Knowledge Management (CIKM), Kansas City, Missouri, pp. 94-101, 1999.

[26] Embley, D. W., Campbell, D. M., Jiang, Y. S., Liddle, S. W., Kai Ng, Y., Quass, D. and Smith, R. D., Conceptual-model-based data extraction from multiple-record Web pages. Data and Knowledge Engineering, 31(3): 227-251, 1999.

[27] Sarawagi, S., Automation in information extraction and integration, Tutorial of The 28th International Conference on Very Large Data Bases (VLDB), 2002.

[28] Kuhlins, S and Tredwell, R. Toolkits for generating wrappers, Net.ObjectDays 2002: Objects, Components, Architectures, Services and Ap-plications for a Networked World, http://www.netobjectdays.org/, LNCS 2591, 2002.

[29] Elmasri, R. and Navathe, S. B. Fundamentals of Database Systems, 4th Ed. Addison Wesley, 2003.

[30] Hsu, C.-N. and Chang, C.-C. Finite-State Transducers for Semi-Structured Text Mining. In Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications, Stockholm, Sweden, 1999. Page 38-49.

[31] Chang, C.-H. and Lui, S.-C., IEPAD: Information extraction based on pattern discovery. Proceedings of the Tenth International Conference on World Wide Web (WWW), Hong-Kong, pp. 223-231, 2001.

[32] Chang, C.-H. and Kuo, S.-C. OLERA: A semi-supervised approach for Web data extraction with visual support. IEEE Intelligent Systems, 19(6):56-64, 2004.

[33] Hogue, A. and Karger, D. Thresher: Automating the Unwrapping of Semantic Content from the World Wide. Proceedings of the 14th International Conference on World Wide Web (WWW), Japan, pp. 86-95, 2005.

[34] Yang, G., Ramakrishnan, I. V. and Kifer, M. On the complexity of schema inference from Web pages in the presence of nullable data attributes, Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM), pp. 224-231, 2003.

[35] Wang, J. and Lochovsky, F. H., Wrapper induction based on nested pattern discovery. Technical Report HKUST-CS-27-02, Dept. of Computer Science, Hong Kong, U. of Science & Technology, 2002.

[36] Wang, J. and Lochovsky, F. H., Data extraction and label assignment for Web databases, Proceedings of the Twelfth International Conference on World Wide Web (WWW), Budapest, Hungary, pp. 187-196, 2003.

[37] Arasu, A. and Garcia-Molina, H., Extracting structured data from Web pages. Proceedings of the ACM SIGMOD International Conference on Management of Data, San Diego, California, pp. 337-348, 2003.

[38] Liu, B., Grossman, R. and Zhai, Y., Mining data records in Web pages. KDD, 601-606, 2003.

[39] Zhai, Y. and Liu, B. Web Data Extraction Based on Partial Tree Alignment. Proceedings of the 14th International Conference on World Wide Web (WWW), Japan, pp. 76-85, 2005.

[40] Liu, B. and Zhai, Y., NET – A System for Extracting Web Data from Flat and Nested Data Records. WISE 2005, 487-495, 2005.

[41] Lan Yi, Bing Liu, and Xiaoli Li. "Eliminating Noisy Information in Web Pages for Data Mining." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003), Washington, DC, USA, August 24 - 27, 2003.

[42] Zhao, H., Meng, W., Wu, Z., Raghavan, V., and Yu, C. Fully Automatic Wrapper Generation For Search Engines. Proceedings of the 14th International Conference on World Wide Web (WWW), Japan, pp. 66-75, 2005.

[43] Lerman, K., Getoor, L., Minton, S. and Knoblock, C. A., Using the structure of Web sites for automatic segmentation of tables. SIGMOD Conference, 119-130, 2004.

[44] Pinto, D., McCallum, A., Wei, X. and Croft, B. C., Table extraction using conditional random fields. SIGIR, 235-242, 2003.

[45] Hsu, C.-N., Chang, C.-H., Hsieh, C.-H., Lu, J.-J. and Chang, C.-C. Reconfigurable Web Wrapper Agents for Biological Information Integration, JASIST (SCI expanded), Special Issue on Bioinformatics, Vol. 56, No. 5, pp. 505--517, 2005.

[46] He, B., Chang, K. C. and Han, J. Discovering complex matchings across web query interfaces: a correlation mining approach. Proceedings of the tenth International Conference on Knowledge Discovery and Data Mining (KDD), pp. 148-157, 2004.

[47] Wu, W., Yu, C., Doan, A. and Meng, W. An interactive clustering-based approach to integrating source query interfaces on the deep web. Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France. pp. 95-106, 2004.

Chia-Hui Chang Chia-Hui Chang is an associate professor at National Central University in Taiwan. She received her B.S. in Computer Science and Information Engineering from National Taiwan University, Taiwan in 1993 and Ph.D. in the same department in Jan. 1999. Her research interests include Web information integration, knowledge discovery from databases, machine learning, and data mining.

Mohammed Kayed is an assistant lecturer at Beni-Suef Universiy. He received the BSc degree from Cairo University, Egypt, in 1994, and the MSc degree from Minia University, Egypt, in 2002. His research interests include information retrieval and Web data extraction. He is also a Ph. D. student at Beni-Suef University. His thesis research concerns on developing a system for Web data extraction.

Moheb R. Girgis is a member of the IEEE Computer Society. He received the BSc degree from Mansoura University, Egypt, in 1974, the MSc degree from Assuit University, Egypt, in 1980, and the PhD from the University of Liverpool, England, in 1986. He is an associate professor at Minia University, Egypt. His research interests include software engineering, information retrieval, genetic algorithms, and networks.

Dr. Khaled F. Shaalan is an assistant professor at the Institute of Informatics, British University in Dubai (BUiD). Before joining BUiD, Khaled lectured at the Faculty of Computers & Information, Cairo University. He is Honorary Fellow, University of Edinburgh, UK. Both his teaching and research are related to language engineering and knowledge engineering.