

ODE: Ontology-Assisted Data Extraction

WEIFENG SU

BNU-HKBU United International College and Shenzhen Key Laboratory of
Intelligent Media and Speech, PKU-HKUST Shenzhen Hong Kong Institution

JIYING WANG

City University of Hong Kong

and

FREDERICK H. LOCHOVSKY

The Hong Kong University of Science and Technology

Online databases respond to a user query with result records encoded in HTML files. Data extraction, which is important for many applications, extracts the records from the HTML files automatically. We present a novel data extraction method, ODE (Ontology-assisted Data Extraction), which automatically extracts the query result records from the HTML pages. ODE first constructs an ontology for a domain according to information matching between the query interfaces and query result pages from different Web sites within the same domain. Then, the constructed domain ontology is used during data extraction to identify the query result section in a query result page and to align and label the data values in the extracted records. The ontology-assisted data extraction method is fully automatic and overcomes many of the deficiencies of current automatic data extraction methods. Experimental results show that ODE is extremely accurate for identifying the query result section in an HTML page, segmenting the query result section into query result records, and aligning and labeling the data values in the query result records.

Categories and Subject Descriptors: H.3.m [Information Storage and Retrieval]: Miscellaneous

General Terms: Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Domain ontology, label assignment, data value alignment

ACM Reference Format:

Su, W., Wang, J., and Lochovsky, F. H. 2009. ODE: Ontology-assisted data extraction. *ACM Trans. Database Syst.*, 34, 2, Article 12 (June 2009), 35 pages.
DOI = 10.1145/1538909.1538914 <http://doi.acm.org/10.1145/1538909.1538914>.

This research was supported by the Research Grants Council of Hong Kong under grant HKUST6172/04E.

Authors' addresses: W. Su, Computer Science and Technology Program, BNU-HKBU-UIC, 28, Jinfeng Road, Tangjiawan Zhuhai, Guangdong Prov., China; email: wfsu@uic.edu.hk; J. Wang, Computer Science Department, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong; email: wangjy@cityu.edu.hk; F. Lochovsky, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong; email: fred@cse.ust.hk.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2009 ACM 0362-5915/2009/06-ART12 \$10.00
DOI 10.1145/1538909.1538914 <http://doi.acm.org/10.1145/1538909.1538914>

1. INTRODUCTION

Databases accessible on the Web, called *Web databases*, compose what is referred to as the *deep Web*. Unlike pages in the surface Web, which are stored for subsequent querying after they are generated, deep Web pages are usually not stored, but are generated dynamically from Web databases in response to a user query submitted through a query interface. A survey in July 2000 estimated that there were 43,000–96,000 deep Web sites and that the deep Web content was 500 times larger than that of the surface Web [Bergman 2001]. A subsequent survey in April 2004 estimated that there were 307,000 deep Web sites [Chang et al. 2004]. In less than 4 years the number of deep Web Sites had expanded 3–7 times.

A Web database responds to a user query with the relevant data, either structured or semistructured, embedded in HTML pages (called *query result pages* in this article). To utilize this data, it is necessary to extract it from the query result pages. Automatic data extraction is very important for many applications, such as metaquerying, data integration, and data warehousing, that need to cooperate with multiple Web databases. Only when the data are extracted and stored in a database can they be easily compared and aggregated using traditional database querying techniques. Consequently, an accurate data extraction method is vital for these applications to operate correctly.

The goal of data extraction is to remove the irrelevant information from a query result page, extract the query result records (referred to in this article as *QRRs*) from the page, and align the data values in the extracted records into a table so that the data values for the same attribute in each record are put into the same column in the table. This process can be viewed as being composed of the following consecutive steps as denoted in Figure 1.

- (1) *Query result section*¹ *identification* decides what section in a dynamically generated query result page contains the data that need to be extracted.
- (2) *Record segmentation* segments the query result section into records and extracts them.
- (3) *Data value alignment* aligns the data values² from multiple records that belong to the same attribute so that they can be arranged into a table.
- (4) *Label assignment* assigns a suitable, meaningful label (i.e., an attribute name) to each column in an aligned table.

Most data extractors assume that a *regular expression* can be employed to model the query result page. Given an alphabet of symbols Σ and a special token “*text*” that is not in Σ , a *regular expression* over Σ is a string over $\Sigma \cup \{text, *, ?, |, (,)\}$ defined as follows [Wang and Lochovsky 2003]:

¹A query result section is also referred to as a data region in the literature.

²We use the term “data value” to refer to a specific value of an attribute in a Web page generated from a Web database. For example, *Irresistable Rise of Harry Potter* may be the data value of the *title* attribute of a book and “15,000” may be the data value of the *price* attribute of a car. We also use the term “raw data string” to refer to *any* string that is not part of a tag element in an HTML file. For example, in a query result section, a raw data string can be a data value or the name of an attribute.

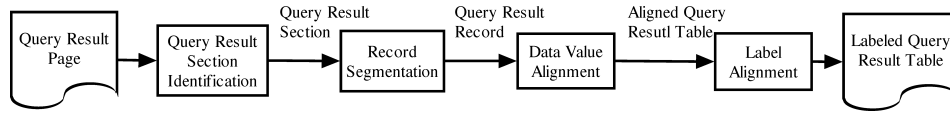


Fig. 1. Data extraction workflow.

- The empty string ε and all elements of $\Sigma \cup \{text\}$ are regular expressions.
- If A and B are regular expressions, then AB , $(A|B)$, and $(A)^?$ are regular expressions, where $(A|B)$ stands for A or B , which means that A and B are *disjunctive* attributes, and $(A)^?$ stands for $(A|\varepsilon)$, which means that A is an *optional* attribute.
- If A is a regular expression, $(A)^+$ is a regular expression, where $(A)^+$ stands for A or AA or ..., which means that A is a *nested data structure*.

Considering the growth and dynamic nature of the deep Web, for data extraction to be effective and practical, each step should be performed *automatically* without requiring any training examples or human involvement. Although there are some existing automatic data extraction systems, as discussed in the related work section, most current data extraction methods suffer from one or more of the following problems.

- (1) *Incapable of processing either zero or few query results.* Almost all existing data extraction methods rely on tag or visual regularity features to do the data extraction. Consequently, they require at least two records in a query result page (e.g., DeLa [Wang and Lochovsky 2003]). However, it is quite common for a query result page to have zero or only one record. Furthermore, if there is only one record, often that record is wrongly split into multiple records if there is some tag or visual regularity within it. According to our survey of 160 Web sites over four ecommerce domains, 84 Web sites return some other QRRs if few or no record satisfies the user's query while the other 76 Web sites do not.
- (2) *Vulnerable to optional and disjunctive attributes.* Optional and disjunctive attributes affect the tag and visual regularity, which may cause data values to be aligned incorrectly.
- (3) *Incapable of processing nested data structures.* Many methods can only process a flat data structure and fail for a nested data structure. However, nested data structures are common on the Web.
- (4) *No label assignment.* While most existing data extraction methods perform the first three data extraction steps, they do not assign a suitable label to each column in the result record table. However, label assignment is important for many applications that need to know the meaning (i.e., the semantics) of the data.

We present a novel data extraction method, ODE (Ontology-assisted Data Extraction), that automatically extracts the QRRs embedded in an HTML page generated by a deep Web site. For a given query, ODE uses both the query interfaces and the query result pages of deep Web sites from the same domain

to automatically construct a domain ontology. In doing so, ODE makes use of the observation that the information from different Web sites can be complementary to each other to help in building the domain ontology. Subsequently, this domain ontology is used in the first, third, and fourth data extraction steps shown in Figure 1. In the query result section identification step, the ontology is used to find a subtree in the HTML tag tree that has a large correlation with the domain ontology. In the data value alignment and label assignment steps, the ontology is used to associate each data value in a QRR with an ontology attribute, with the assistance of a maximum entropy model that assigns attribute names to data values using context and tag structure information as features. To our knowledge, we are the first to use a maximum entropy model for data extraction. ODE is fully automatic and overcomes many of the shortcomings of current automatic data extraction methods. Furthermore, experimental results show that ODE is extremely accurate in identifying the query result section in an HTML page, segmenting the query result section into records, and aligning and labeling the data values in the records.

The rest of the article is organized as follows. Section 2 presents an overview of the ODE data extraction system. Section 3 describes the algorithm for constructing an ontology from the query interfaces and query result pages of a domain. Section 4 shows how to do the data extraction using the ontology. Section 5 reports the experimental results of applying ODE on four popular domains on the Web. Section 6 reviews related work. Finally, section 7 concludes the article.

2. ODE OVERVIEW

To address the data extraction problems enumerated in Section 1, we propose to use an ontology-assisted method. The ontology for a domain is first constructed from query result pages and query interfaces of the Web sites, and then used to extract data records from a query result page in the domain. In this section, we first present the observations about query result pages and query interfaces of Web databases in a domain,³ on which the ontology construction approach is based. Then, examples are given to motivate the intuition behind our ontology-assisted data extraction approach. Finally, the problem formulation and the framework of the ODE method are presented.

2.1 Observations About Deep Web Query Interfaces and Query Result Pages

2.1.1 Amazon Effect. The “Amazon effect” states that Web databases within a domain tend to be influenced by their peers as the number of Web databases grows [Chang et al. 2004]. One consequence of this effect is that the query interface vocabulary of Web databases within a domain tends to converge to a small size. By manually checking 160 query result pages from four popular ecommerce domains, Book, Airfare, Music, and Movie, we found that the

³These observations come from, and therefore are primarily applicable to, consumer-oriented, ecommerce Web databases. Whether, and to what extent, they are applicable to other types of Web databases (e.g., biological Web databases) would be interesting to explore, but is not pursued in this article.

<p>* Author: <input type="text"/></p> <p>* Title: <input type="text"/></p> <p>* ISBN: <input type="text"/></p> <p>* Keywords: <input type="text"/></p> <p>* Publisher: <input type="text"/></p> <p>Published Date: <input type="text"/> min to <input type="text"/> max</p>	<p>Title <input type="text"/></p> <p>Author <input type="text"/></p> <p>Publisher <input type="text"/></p> <p>Category <input type="text" value="-- All Categories --"/></p> <p>Format <input type="text" value="-- All Formats --"/></p> <p>Price <input type="text" value="-- All Price Ranges --"/></p>
---	--

<p>HARRY POTTER AND THE HALF-BLOOD PRINCE by ROWLING, J.K. hardback / BLOOMSBURY PUBLISHING PLC / 20051003 Available: Usually ships within 24-48 hours. ISBN: 0747581428(34.872314453125)</p>	<p><u>Harry Potter and the Half-Blood Prince</u> by <u>Rowling, J.K.</u> Format: Hardcover (Cloth) Price: \$29.99 Published: Arthur A. Levine, 2005 Inventory Status: Usually Ships in 1-5 days</p>
--	---

(a)
(b)

Fig. 2. Fragments of query interfaces and query results. Each column contains a fragment of a query interface and a fragment of a query result from a book selling Web site.

“Amazon effect” also happens in the query result pages within a domain, namely, as follows.

- (1) The size of the set of attribute labels in the query result pages of a domain is not arbitrarily large, but converges to a relatively small size. Thus, the attribute names/labels are usually the same or similar from different Web sites within a domain.
- (2) The representations of the values for the same attribute from different Web sites within a domain are usually similar. For example, the Price representation in the different Web sites in the Book domain in our survey is almost always “\$ \pounds (REAL)”. Thus, an attribute’s data type and value domain in the query result pages are also the same or similar.
- (3) The attribute sequences within a record from different Web sites within a domain are usually similar. For example, a record in the Book domain usually begins with the title, followed by the author, then the format, followed by the publisher, etc., as illustrated by the records in Figure 2.

2.1.2 Complementary Information in Query Result Pages. The Amazon effect focuses on the information similarities in the query interfaces and query result pages among Web databases. However, we found that there are also some *information dissimilarities* among the query result pages from different Web sites within the same domain, which are often complementary to each other for data extraction. For example, it is possible that some data values can get their labels from the query result pages of other Web sites. In the query result fragment in Figure 2(a), the attribute value “hardback” can be labeled as “Format” from the information available in the query result fragment in Figure 2(b).

2.1.3 Complementary Information in Query Interfaces. The information matching (within one site, or among multiple sites) between a query result page and a query interface also can be very useful to understand the query result pages and has been used in label assignment in multiple data extraction

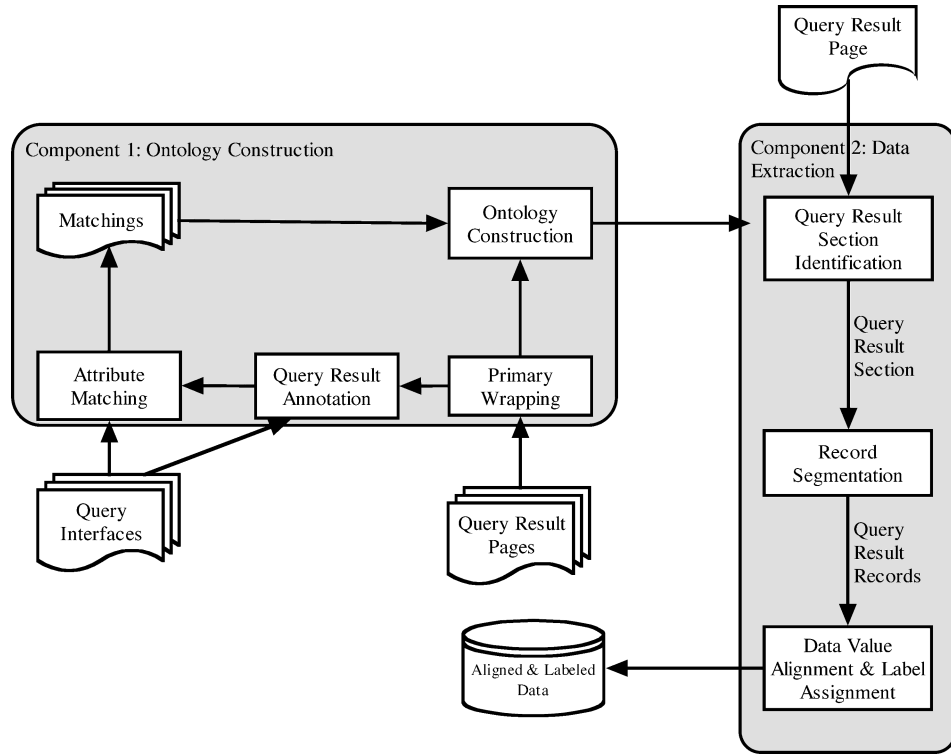


Fig. 3. Ontology-assisted Data Extraction (ODE) workflow.

models [Tijerino et al. 2005]. For example, in the query result fragment in both Figure 2(a) and Figure 2(b), the attribute corresponding to the data value “Harry Potter and the Half-Blood Prince” can be labeled as “title” from their own query interfaces. Moreover, the query interfaces from other Web sites within the same domain can be very useful to understand the query result page. For example, in Figure 2, the attribute corresponding to the data value “2005” in the query result fragment in Figure 2(a) can be labeled as “Published Date” from the information available in the query interface in Figure 2(b).

2.2 ODE Workflow

Based on the observations in Section 2.1, our objective is to acquire sufficient domain knowledge that is available from the query interfaces and the query result pages in a domain and to use the acquired knowledge to extract the data instances from the query result pages of the domain. A domain ontology is used to represent the acquired domain knowledge.

Figure 3 shows the workflow of our ontology-assisted data extraction system, which consists of two main components: ontology construction and data extraction. The ontology construction component (i.e., Component 1) builds the

ontology for the entity⁴ of a domain. The ontology contains the domain knowledge, which consists of the descriptions for all attributes in the domain, extracted from the query interfaces and query result pages of multiple Web databases in the domain. Instead of constructing the ontology manually, we utilize the matching among query interfaces and query result pages within a domain to automatically generate the ontology. After the domain ontology is built, the data extraction component (i.e., Component 2) extracts the data from the query result pages using the ontology for several data extraction substeps as described in detail in Section 4.

2.3 Advantages of ODE

ODE addresses the shortcomings of current data extraction methods enumerated in Section 1 as follows.

- (1) As described in detail in Section 4.2, ODE can effectively process query result pages containing zero or few QRRs.
 - (a) When there is only one QRR in a query result page, ODE identifies the part in the page that has the largest correlation with the ontology to be the only QRR.
 - (b) When there is no QRR in a query result page, ODE finds no section that has a high correlation with the ontology (i.e., no section will be identified as the query result section).
- (2) Since the ontology provides an overview of the QRRs generated from a Web database, the influence of optional and disjunctive attributes on the data extraction is decreased, especially for the data value alignment step. ODE maps the raw data string, which could be an attribute label or some actual data value, to attributes in the ontology according to the various kinds of information found in an HTML page, such as tag or context information. Recall that the ontology is constructed by extracting attributes from query interfaces and query result pages of multiple Web databases in a domain. Thus, in the data value alignment step for each single Web site, it is not really very important whether the raw data strings to align are generated by optional, disjunctive, or “repeated” attributes.
- (3) By mapping the raw data strings in a query result page to the attributes in the domain ontology, we can discover the nested structure within a record when the raw data strings are mapped to the same attributes consecutively.
- (4) With the aid of the ontology, each aligned column can be assigned a suitable name from the attribute description in the ontology.

Furthermore, ODE also has the following additional advantages.

- (1) Fully automatic data extraction: All data extraction steps can be performed automatically (i.e., no manual labeling or other human interaction is required).

⁴The entity of a domain is the major concept that is described by the data of the domain. For example, book, car, and movie might be the entities in domains dealing with book sales, automobile sales/rentals, and movie sales/rentals, respectively.

- (2) Understanding the query result page semantically: By assigning a suitable label to each column of the extracted table, the meaning of the data can be more easily interpreted (i.e., understood semantically), which can be very useful for many other applications (e.g., data integration and data warehousing).

3. ONTOLOGY CONSTRUCTION COMPONENT




The goal of the ontology construction component is to build an ontology for a domain using the query interfaces and some training query result pages from Web sites within the domain. As shown by Component 1 of Figure 3, the workflow for the ontology construction component consists of the following four steps.

- (1) Given a set of query result pages within the same domain, the *primary wrapping* step uses a primary wrapper to extract the QRRs from the query result pages and align them into a query result table.
- (2) The *query result annotation* step uses a query result annotator to assign a suitable name to each column of the output table of the primary wrapper, using information obtained from the query result pages and query interfaces.
- (3) The *attribute matching* step matches the columns of the query result tables and query interfaces to each other using an attribute matcher.
- (4) Finally, the *ontology construction* step uses the matching results to construct the ontology.

In this section, the first subsection presents the domain-ontology specification. Then each subsequent subsection describes a step in the ontology construction workflow in detail.

3.1 Ontology Specification

We define a *domain ontology* to be a data model that describes a set of concepts within the domain and the relationships between those concepts. It should be understandable to a computer so that it can be used to reason about the objects within that domain. In particular, the ontology O for a domain is composed of two related components.

- (1) An *ontology attribute model* that describes the organization of the attributes of the entity within the domain. For example, Figure 4 shows the ontology attribute model for the Book domain ontology. A solid border rectangle  denotes the entity of the domain (e.g., Book). Each dashed border rectangle  represents an attribute of the entity (e.g., Title, Price, Subject, Author, Last Name, First Name, and Publisher). The solid triangle  defines an aggregation with the superpart attribute connected to the apex of the triangle and the component-part attributes connected to its base. For example, Author is an aggregation of Last Name and First Name in Figure 4.
- (2) An *attribute description* that lists the values for each attribute field in the ontology O . As shown by the example attribute description for the attribute Price in the Book domain in Figure 5, each attribute in O contains the following fields.

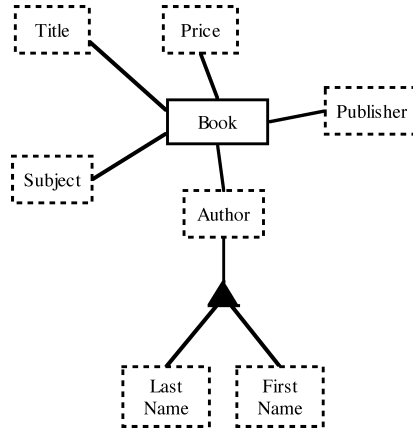


Fig. 4. Ontology attribute model for the Book domain.

- (a) *Name* is the name of the attribute (e.g., Price).
- (b) *Alias* is an alias for the attribute. There can be multiple aliases for each attribute (e.g., our Price, lowest Price).
- (c) *Data type* is the data type of the attribute, which is obtained from the parsing of its values (e.g., Real). In our experiments, the data type can be datetime, int, real, price, and string. We empirically set an attribute to a certain data type if 80% of its values in the training Web sites belong to that data type.
- (d) *External representation* is the representation of the attribute value in a query result page. There can be multiple representations of the values for each attribute (e.g., \$, £).
- (e) *Value* contains the attribute values that have appeared in the training query result pages (e.g., \$5.10, £12.20, \$16.30).
- (f) *Value probability* is the likelihood that the attribute value occurs in a record (e.g., 88.5%).
- (g) *Name probability* is the likelihood that the name or one of its aliases occurs in a record in a query result page (e.g., 78.2%).
- (h) *Max occurrence* is the maximum number of occurrences of the attribute's values within a record in the training query result pages. It is a positive integer (e.g., 1).

3.2 Primary Wrapping

The domain ontology is built from the query interfaces and the query result records (i.e., instances of the ontology) within the same domain. However, since obtaining the query result records is the goal of data extraction, it appears that we would have an unsolvable problem. To overcome this problem, a *primary wrapper* is used to obtain *some* instances from a query result page. The requirement for the primary wrapper is that it can correctly identify the QRRs

Price
Name: Price
Alias: our Price, lowest Price
Data Type: price
Value: \$5.10, £12.20, \$16.30
External Representation: \(\$\£)? (Real)⁵
Value probability: 88.5%
Name Probability: 78.2%
Max Occurrence: 1
End

Fig. 5. Ontology description for the attribute Price in the Book domain.

in most cases if there are multiple QRRs in a query result page.⁶ Most existing wrapper construction methods satisfy this requirement. In our experiments, the PADE wrapper construction method is used as the primary wrapper [Su et al. 2009].

PADE builds a wrapper for a query result page from its HTML tag tree. Similar to Simon and Lausen [2005] and Zhai and Liu [2006], PADE assumes that similar QRRs are represented as child subtrees of the same parent node in a tag tree. The similar QRRs form a data region. To identify a data region, PADE requires that the data region include several sets of *similar* nodes in the tag tree. Two nodes n_1 and n_2 are similar if their similarity, sim_{12} , is larger than or equal to a threshold T_{nsim} . For n_1 and n_2 , their corresponding tag strings s_1 and s_2 are used to evaluate their similarity using the normalized edit distance between them [Baeza-Yates 1989; Gusfield 1997]. To overcome the problem caused by optional attributes and repetitive subparts, the similarity threshold T_{nsim} is empirically set to 0.6 [Zhai and Liu 2006]. However, it is still possible that two similar nodes have a normalized edit distance less than a small threshold because of the repetitive subparts. Therefore, the so called *tandem repeats*⁷ that are present in both tag strings are identified and zero cost is allowed for deletion and insertion operations inside additional repetitive instances [Simon and Lausen 2005].

Given that a query result page contains at least two QRRs, PADE adopts the solution used in Simon and Lausen [2005] and Zhai and Liu [2006] to identify data regions. Starting from the root, it searches the tag tree for similar children nodes under the same parent and the ordered sets of those similar nodes compose the raw data regions. Next, PADE segments the raw data regions into raw data records according to the starting and ending positions of the tandem repeats and the visual gap between the segmented records. Finally, but different from Simon and Lausen [2005] and Zhai and Liu [2006], PADE tries to compare the segmented records and merge the raw data regions containing similar records, even though they may be present under different parent nodes.

⁵The symbol “\” indicates that “(\$\£)” should be treated as characters while the symbol “?” indicates that the characters “(\$\£)” are optional.

⁶Note that the primary wrapper may still have some or all of the deficiencies enumerated in Section 1. However, this is not a problem as the primary wrapper just needs to be “good enough” to allow the ontology to be constructed from its extraction result.

⁷A *tandem repeat* refers to repeated tag substrings that are directly adjacent to each other.

In general, after performing the data region merge step, there may still be multiple data regions in a query result page. However, PADE assumes that there is only one actual query result section in a query result page that contains the QRRs. To determine which of the remaining data regions corresponds to the actual query result section, PADE uses the following three simple heuristics.

- (1) *The query result section is usually located at the center of the query result page* [Zhao et al. 2005]. For each remaining data region, a center distance is calculated between the center of the page and the center of the data region. For each data region d , a center distance weight, which is calculated as the smallest center distance divided by d 's center distance, is assigned for d .
- (2) *The query result section usually occupies a large space in the query result page* [Zhao et al. 2005]. For each remaining data region d , an area weight, which is calculated as d 's area divided by the largest area data region, is assigned for d .
- (3) *Each QRR usually contains more raw data strings than the records in other sections.* For each remaining data region d , a value weight, which is calculated as the average number of raw data strings in a record divided by the largest average number of data values, is assigned for d .

The aforesaid three weights are summed and the data region that has the largest summed weight is selected as the query result section. Records in this data region are assumed to be the QRRs.

After identifying the QRRs, PADE performs data value alignment using a pair-wise alignment to align the data values in the QRRs so that they can be put into a table. Both data content similarity and tag structure are used in the alignment process. In particular, given a set of QRRs, every two different records are aligned first. Since the data values for the same attribute usually have the same data type and are often similar to each other, the pair-wise record alignment consists of finding an alignment with the maximal content similarity score. After all pairs of records are aligned, an iterative alignment is performed to align all records, in which the goal of each step is to find a global alignment that has the largest number of pair-wise alignments. Finally, a nested data structure identification method is used to handle any nested structure that exists in the QRRs.

Experimental results show that PADE is very accurate in extracting and aligning the QRRs in a query result page, achieving an accuracy of 96% in extracting and aligning the QRRs when there are at least two QRRs in the query result page. While PADE suffers from the first and fourth deficiencies listed in Section 1, nevertheless, the high accuracy of PADE provides a solid basis for building a good ontology. Table I shows the output tables constructed for the query result segments in Figure 6. PADE can also effectively handle the optional attribute and nested structure problems. For example, the optional attribute, ISBN, can be identified, even though it does not occur in the first record. Furthermore, there are two authors in the first record (i.e., the occurrence of author is 2), and these two occurrences, which form a nested structure, are correctly identified and aligned.

Visions of the East: Orientalism in Film
 by Matthew Bernstein & Gaylyn Studlar
 Published/Distributed by Rutgers University Press
 \$25.35

Orientalism: A Reader
 by Alexander Lyon Macfie
 Published/Distributed by NYU Press
 ISBN No: 0814756654
 \$22.50

Orientalism in Art
 by Christine Peltre
 Published/Distributed by Abbeville Press
 ISBN No: 0789204592
 \$62.50

Fig. 6. Query result segment example.

Table I. Aligned Result for Figure 6

Visions of the East: Orientalism in Film	by Matthew Bernstein Gaylyn Studlar	Published/Distributed by Rutgers University Press		\$25.35
Orientalism: A Reader	by Alexander Lyon Macfie	Published/Distributed by NYU Press	ISBN No: 081475 6654	\$22.50
Orientalism in Art	by Christine Peltre	Published/Distributed by Abbeville Press	ISBN No: 078920 4592	\$62.50

Table II. Annotated Query Result table for Figure 6

Title	by	Published/Distributed by	ISBN No:	Price
Visions of the East: Orientalism in Film	Matthew Bernstein Gaylyn Studlar	Rutgers University Press		\$25.35
Orientalism: A Reader	Alexander Lyon Macfie	NYU Press	0814756654	\$22.50
Orientalism in Art	Christine Peltre	Abbeville Press	0789204592	\$62.50

3.3 Query Result Annotation

To assign labels to the columns of the data table containing the extracted QRRs (i.e., to interpret the meaning of the data values), the four heuristics proposed in Wang and Lochovsky [2003] are used. Table II shows the annotated query result table for the Book records in Figure 6.

Heuristic 1: Match query interface element labels to data values. Assuming that the Web database designers try their best to answer user queries with the most relevant data, then keywords submitted through one specific query interface element will reappear in the corresponding data values in the query result pages. Therefore, for each keyword query, if the keyword mostly appears in one specific column of the result data table, then we assign the label of that query interface element to the column. For example, given the keyword query *title="orientalism"*, the first column in Table II is labeled "Title" according to this heuristic.

Heuristic 2: Search for voluntary labels in table headers. The HTML specification defines some tags such as <TH> and <THEAD> for page designers to voluntarily list the headings for the columns of their HTML tables [World Wide Web Consortium 1999]. Moreover, these labels are usually placed nearby the data values in a query result page. Therefore, the HTML code near (usually at the top of) the data values is examined for possible voluntary labels.

Heuristic 3: Search for voluntary labels encoded together with data values. Some query result pages encode the labels of data values together with the data values. Therefore, for each column of the data table we try to find the maximal prefix and maximal suffix shared by all cells of the column and assign the meaningful prefix to that column and the meaningful suffix to the column next to that column as the labels of those columns. For example, the second, third, and fourth columns in Table II get their label using this heuristic. In the fourth column of Table I, the maximal prefix shared by all cells of the column is “Published/Distributed by,” which is used as the label for that column.

Heuristic 4: Label data values in conventional formats. Some data have a conventional format (e.g., a date is usually organized as “dd-mm-yy”, “dd/mm/yy”, etc., an email address usually contains the symbol “@”, price usually has the symbol “\$”, “£”, etc.). Thus, such information is used to label the corresponding data values. For example, the fifth column in Table II is labeled “Price” using this heuristic.

If there are conflicts among these heuristics, the heuristic priority sequence is Heuristic 1 > Heuristic 2 > Heuristic 3 > Heuristic 4. Since the query interface elements and the data values may not be perfectly matched, some columns may not have any labels. Nevertheless, experimental results in Wang and Lochovsky [2003] show that these four heuristics achieve a label assignment accuracy of around 80%.

3.4 Attribute Matching

Given the query interfaces and the aligned tables for the query result pages from Web sites within the same domain, an attribute matcher identifies the matchings among columns from the query result tables and query interfaces from different Web sites. Three kinds of matchings, which are explained in the following subsections, are identified.

3.4.1 Value-Level Matching. Value-level matching identifies the matchings among the query result table columns from different Web sites using the data value similarity between different Web sites, which is based on the Amazon effect in the query result pages described in Section 2.1.1. Value-level matching can be viewed as an instance-based schema matching step and any existing instance-based schema matching method can be used. In our experiments, the method in Bilke and Naumann [2005] is used to identify the 1:1 simple matchings among different query result tables. We first discover duplicates among different tables, which refer to multiple representations of the same real-world object, and then use these duplicates’ attribute values to identify the matching columns among different tables.

Table III. Identified Matchings for Book Domain

Matching #	Discovered Matching
1	Author (String) = {Last Name (String), First Name (String)}
2	Category (String) = Subject (String)
3	Title (String)

3.4.2 Label-Level Matching. Label-level matching identifies the matchings among the query result table columns from different Web sites according to the label cooccurrence pattern in the query result tables. Both simple matchings (i.e., 1:1 matching) and complex matchings (i.e., 1: n or $n:m$ matchings) are identified. A holistic matching algorithm,⁸ such as He and Chang [2006], Su et al. [2006], is applied to the labels of the query result tables after the value-level matching, according to the label cooccurrence pattern, without requiring any domain knowledge. For example, the first and second matchings in Table III are the matchings identified for the Book domain using label-level matching.

3.4.3 Label-Value Matching. Label-value matching identifies the matchings among query interfaces and query result tables among Web sites (i.e., inter-Web site). A probing method, such as in Gravano et al. [2003], Wang et al. [2004], is used for this task. A query submission component exhaustively submits instances into the input elements in the query interfaces and the query results are collected and extracted into a data table. The reoccurrence of submitted queries in the columns of this table are counted and stored into a Query Occurrence Cube (QOCube). The QOCube is aggregated/projected in different ways to reflect the relationship between pairs of schemas (intrasite and inter-site). The label-value matching bridges the value-level and label-level matchings.

After the preceding three matchings are performed, the matching results can be used to deal with the following tasks.

- (1) *Checking the assigned label.* If a column c_1 with label l_1 is not matched to most columns from other Web sites whose label is also l_1 , we think that the label l_1 is not correctly assigned to c_1 and replace l_1 with the most common label for columns to which c_1 matched.
- (2) *Assigning a suitable label for columns without an assigned label.* If a column c_1 with no label assigned yet is matched to another column c_2 from a different Web site with a label l_2 , l_2 is assigned to c_1 as its label.
- (3) *Matching conflict resolution.* If the column c_1 matched to c_2 and c_3 , but c_2 and c_3 are not matched, we assume one of the two matchings is incorrect. Suppose that c_2 occurs in the query result pages of m Web sites and c_3 occurs in the query result pages of n Web sites and $m > n$. We will further assume that the matching between c_1 and c_2 is correct and the matching between c_1 and c_3 is incorrect because the matchings among attributes with higher occurrence, which are identified by the aforesaid three matching identification methods, are more accurate.

⁸A holistic matching algorithm considers all inputs simultaneously to do the matching rather than doing the matching, say, in a pair-wise manner.

Table III shows some identified label matchings for the Book domain. Matching 1 denotes that the label “Author” with data type String is equal to a combination of label “First Name” and “Last Name,” both with data type String. For an attribute that does not match to any other attribute, we consider it to be a *unique matching*, such as Matching 3 in Table III, so that all attributes can be handled in a uniform way. Experimental results show that the attribute-matching step increases the annotation accuracy to 90% on average.

3.5 Ontology Construction

The attributes for a domain ontology are created by organizing the matchings identified by the attribute-matching step as follows.

- (1) If the matching is unique, an attribute a is created. For example, given the unique matching “Title (String)”, an attribute with name *Title* and data type “string” is created.
- (2) If the matching is 1:1, an attribute is created in which any label in the matching is randomly selected as the attribute name and other labels are set as aliases. For example, given the matching “Category (String) = Subject (String)”, an attribute *Category* is created with data type “string” and alias “Subject”.
- (3) If the matching is 1: n , $n + 1$ attributes are created. For example, given the matching “Author (String) = {Last Name (String), First Name (String)}”, three attributes, *Author*, *First Name*, and *Last Name*, are created with data type “string”. As explained in Section 3.1, for such a matching *Author* is connected to the apex of a solid triangle \blacktriangle and *Last Name* and *First Name* are connected to the base of the solid triangle \blacktriangle , denoting that the strings generated by *Last Name* and *First Name* can be concatenated to be an *Author* string.
- (4) If the matching is $n:m$ $\{a_1, \dots, a_n\} = \{a_{n+1}, \dots, a_{n+m}\}$, we can treat it as a 1: n matching and a 1: m matching by generating an artificial attribute a_a and two matchings to this attribute: $a_a = \{a_1, \dots, a_n\}$ and $a_a = \{a_{n+1}, \dots, a_{n+m}\}$ and process the 1: n matching and 1: m matching using the method described in the 3rd point given before. Hence $n + m + 1$ attributes are created. For example, given a 2:2 matching $\{\text{area, town}\} = \{\text{city, location}\}$ in the Hotel domain, which describes the location of a hotel, we divide the matching into two 1:2 matchings $a_i = \{\text{area, town}\}$ and $a_i = \{\text{city, location}\}$. Each of the two 1:2 matchings is then handled using the method for the 1: n matching described previously.
- (5) The values of an attribute are obtained from the data values that appear in the query result pages from the training Web sites.
- (6) The data type and external representation of an attribute are derived through the parsing of its values. The parsing starts from some conventional data format, such as datetime, int, real, etc., and then tries to find the maximal prefix and maximal suffix shared by data values with the same label from the same domain. In our implementation, the data types parsed include datetime, int, real, price, and string. The data type can be

obtained through analyzing attribute values and the pattern or format of the values, for example, \$300 for *price* and 3:00PM for *datetime*. When the value type is difficult to determine, a default value type (i.e., *string*) is used. For example, given data values “Color: Red” and “Color: Blue”, the external representation is “color: [string]”.

- (7) The value probability of an attribute is calculated as the Web-site-average of the average occurrence probability of a data value in a QRR within each training Web site.
- (8) The name probability of an attribute is calculated as the Web-site-average of the average occurrence probability of the attribute name in a QRR within each training Web site.
- (9) The max occurrence is the maximum occurrence count of the attribute value in all QRRs of the training Web sites.

4. DATA EXTRACTION COMPONENT

After constructing the domain ontology, we use it to perform the data extraction. As shown by Component 2 of Figure 3, the workflow of the data extraction component is composed of the following three steps.

- (1) *Query result section identification*: Since the query result section is a section whose text content is mostly related to the domain knowledge, we identify the query result section by constructing a tag tree from the query result pages and finding the subtree whose raw data strings have a large correlation with the ontology.
- (2) *Record segmentation*: To segment the query result section into query result records (QRRs), we use the same techniques as for the primary wrapper described in Section 3.2.
- (3) *Data value alignment and label assignment*: Both tasks can be performed simultaneously by assigning an attribute name of the ontology to each data value of a QRR. A maximum entropy model is used to do attribute name assignment for data values.

In the rest of this section, we first briefly review the maximum entropy model. Then each step of the data extraction component is discussed in detail.

4.1 Maximum Entropy Model

The maximum entropy model has been successfully applied to many problems, such as part-of-speech tagging [Ratnaparkhi 1996], name entity recognition [Borthwick 1999], and text categorization [Zhang and Oles 2001]. Intuitively, the principle of maximum entropy is simple: We model all that is known and assume nothing about that which is unknown. In other words, given a set of constraints, we choose a model that is consistent with all the constraints, but otherwise is as uniform as possible.

The goal of a maximum entropy model is to construct a stochastic model that accurately represents the behavior of a random process [Berger et al. 1996]. Such a model is a method of estimating the conditional probability that, given

a context x , the process will output y . We denote a single observation by y , a random variable that takes on values in alphabet Y . Since the identity of y is influenced by some conditioning information $x \in X$, we seek a conditional model $p(y|x)$. In ODE, x can be any text or tag information and y can be any observation relevant to x . For example, if we observe that the attribute Author usually follows the attribute Title in the Book domain, then we assign x to be “Title as previous attribute” and y to be “Author”. To study the random process, we observe its behavior for a while, collecting some training samples (x_1, y_1) , (x_2, y_2) , \dots , (x_n, y_n) . As a particular pair (x, y) may occur in some or all of the samples, the training samples are summarized in terms of their empirical probability

$$\tilde{p}(x, y) \equiv \tilde{c}(x, y) / \sum_{x, y} \tilde{c}(x, y), \quad (1)$$

in which $\tilde{c}(x, y)$ is the number of times (x, y) appears in the collection.

Given a set of features, which determines the statistics that we feel are important in modeling the process, we would like to build our model in accordance with these statistics. Generally a feature f_i can be defined as

$$f_i(x, y) = \begin{cases} 1 & \text{if the feature is expressed in case } (x, y). \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The expected value of feature f_i with respect to the empirical distribution $\tilde{p}(x, y)$ is

$$\tilde{p}(f_i) = \sum_{x, y} \tilde{p}(x, y) f_i(x, y), \quad (3)$$

while the expected value of feature f_i with respect to the model $p(y|x)$ is

$$p(f_i) = \sum_{x, y} \tilde{p}(x) p(y|x) f_i(x, y), \quad (4)$$

in which $\tilde{p}(x)$ is the empirical distribution of x in the training samples.

The maximum entropy model seeks to maximize the entropy $H(p)$ with respect to $p \equiv p(y|x)$ with the constraints $\tilde{p}(f_i) \equiv p(f_i)$ for all features $i = 1, \dots, n$. To accomplish this, a parameter λ_i (the Lagrange multiplier) is generated for each feature f_i by one of many iterative algorithms, such as Generalized Iterative Scaling or Conjugate Gradient Ascent [Minka 2003]. The joint probability of a process is determined by those parameters whose corresponding features are active (i.e., those λ_i such that $f_i(x, y) = 1$).

The context and tag structure in a query result page provide useful information that can be used as features for the different steps of data extraction. The context information includes the following in our experiments.

- (1) Is the raw data string the name, alias, or the value of an attribute in the ontology?
- (2) What is the possible attribute of the previous two and subsequent two raw data strings?
- (3) Is there at most one occurrence for a raw data string in the current QRR?

- (4) What is the possible data type candidate of the raw data string?
- (5) What is the external representation of the raw data string?
- (6) Is this the first raw data string for the current QRR?
- (7) Is this the last raw data string for the current QRR?
- (8) How often does an attribute value occur in a record?
- (9) How often does the attribute name or one of its aliases occur in a record?

The tag structure information includes the following information.

- (1) Does the raw data string have the same tag path as previous raw data strings in the current QRR? What is the label of those raw data strings with the same tag structure?
- (2) Does the raw data string have the same tag structure as one of the raw data strings in the previous QRR? What is the label of those raw data strings with the same tag structure?

Each of the preceding questions and their answers can be encoded as a feature. For example, a useful feature in the Book domain, based on the second context information question, can be

$$f(x, y) = \begin{cases} 1 & \text{if } x = \text{"Title as previous attribute"} \text{ and } y = \text{"Author"} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

This feature helps the model to discover the fact that “Author” usually follows “Title” in a book instance. Therefore, the parameter corresponding to this feature will hopefully boost the probability $p(d_i, \text{"Author"})$ if the label of the data value d_{i-1} is “Title”.

4.2 Query Result Section Identification

Query result section identification recognizes the query result section in a query result page, which is denoted as the first block in Component 2 of Figure 3. ODE first performs query result section identification using the primary wrapper PADE. As discussed in Section 3.2, PADE is very precise at identifying a query result section if two or more QRRs exist in the query result page. For QRRs that are in the same format and are separated in different regions, PADE can also effectively combine them into the same query result section. However, ODE still can improve on PADE by effectively judging whether the instances in the query result section returned by PADE are the actual QRRs or not. We assume that an instance is a QRR if its *correlation* with the ontology O is larger than a threshold T_c . To allow the threshold T_c to be adaptive to the domain, it is empirically set to be half of the smallest correlation with the ontology O for all QRRs in the training Web sites.

If the query result section returned by PADE does not contain QRRs, we assume one of the following cases.

- (1) PADE incorrectly identified the query result section when there are more than two QRRs in the page. This may happen in some pages when the query result section is relatively small compared to other sections, such as advertisements or announcements, or when the query result section is not in the center of the page.

- (2) PADE failed to identify the query result section when there is only one QRR in the query result page.
- (3) There is truly no QRR in the query result page.

To deal with the previous cases, ODE carries out the following steps in sequence.

- (1) Determine whether there is another query result section that contains QRRs and return the query result section if it exists. ODE removes the section that has been incorrectly identified as the query result section from the query result page and identifies a new section in the remaining query result page using PADE. If the newly identified section includes QRRs whose correlation with the ontology O is larger than the threshold T_c , ODE then returns the query result section; otherwise, the following will happen.
- (2) ODE assumes that there is only one QRR in the query result page and finds the subtree with the largest correlation with the ontology, using the algorithm described in Section 4.2.2. If the correlation between the subtree and the ontology O is larger than T_c , ODE returns the subtree as a single QRR; otherwise, we have the following.
- (3) ODE assumes that there is no QRR in the page.

Consequently, a good instance-ontology correlation calculation method is vital for correctly identifying the query result section.

4.2.1 Instance-Ontology Correlation. The instance-ontology correlation calculation is used to evaluate the similarity between an instance $D = \{d_1, \dots, d_n\}$, which may or may not be a QRR, and an ontology O . For each raw data string d_i of D , its weight S_i for the correlation is defined to be

$$S_i = \begin{cases} p_{nj} & \text{if } d_i \text{ is the name or alias of attribute } a_j \\ p_j & \text{otherwise and } a_j \text{ is the largest probability attribute that } d_j \text{ can be.} \end{cases} \quad (6)$$

That is, if d_i is identified to be the name of an attribute a_j , its weight is equal to the name probability p_{nj} of a_j . Otherwise, if d_i is recognized to be a value of several attributes, its weight is equal to the probability p_j of attribute a_j that has the largest probability and the maximum occurrence constraint of a_j has not been violated. The normalized sum of all the S_i (for d_i to d_n) is then calculated to be the correlation between D and O

$$\text{Corr}(D, O) = \frac{\sum_{i=1}^n S_i}{\sqrt{mn}}, \quad (7)$$

in which m is the number of attributes in O and n is the number of raw data strings in D .

4.2.2 Maximum-Correlation-Subtree Identification. According to the nested structure of the HTML tags in the source HTML file, each query result page naturally forms a tag tree. The root of the tree is the `<HTML>` tag and all content

```

Function MaximumCorrelationSubtreeIdentification( $R, O$ )
1.  $T \leftarrow R$ 
2. while ( $T \neq \text{leaf node}$ )
3.    $T_i \leftarrow$  a child of  $T$  that has the largest correlation with  $O$ 
4.   if ( $\text{Corr}(T_i, O) > \text{Corr}(T, O)$ )
5.      $T \leftarrow T_i$ 
6.   else break
7. endwhile
8. while (true)
9.    $T_l \leftarrow$  the immediate left sibling of  $T$ 
10.  if ( $\text{Corr}(T_l + T, O) > \text{Corr}(T, O)$ )
11.     $T \leftarrow T_l + T$ 
12.  else break
13. endwhile
14. while (true)
15.    $T_r \leftarrow$  the immediate right sibling of  $T$ 
16.   if ( $\text{Corr}(T_r + T, O) > \text{Corr}(T, O)$ )
17.      $T \leftarrow T_r + T$ 
18.   else break
19. endwhile
20. return  $T$ 

```

Fig. 7. Maximum correlation subtree identification.

nodes, such as text and images, are leaf nodes. Figure 7 shows the algorithm to identify the subtree S in a tag tree that has the largest correlation with the ontology O . Starting from the root R , a top-down loop is first used to identify a subtree T which has the largest correlation score with the ontology (lines 1–6). Assuming a subtree T is an instance of a QRR, its correlation $\text{Corr}(T, O)$ with the ontology O is calculated using Eq. (7) given in Section 4.2.1. The loop stops when each of the children of T has a correlation score smaller than T with the ontology O . Moreover, T is combined either with its immediate left sibling node if this combination increases the correlation with the ontology (lines 8–13), or with its immediate right sibling node if this combination increases the correlation with the ontology (lines 14–19).

4.3 Record Segmentation

The record segmentation algorithm, which is denoted as the second block in Component 2 of Figure 3, segments the nodes in a query result section into a set of records given the query result section S . If S contains only one QRR, the record segmentation does nothing. If S contains more than one QRR, the record segmentation segments S into multiple QRRs. Given a query result section S , we first find continuous repeated patterns (*C-repeated patterns*), each of which is a repeated substring of S having at least one pair of its occurrences that are adjacent [Wang and Lochovsky 2003]. For example, in a query result section $R=ABABABA$, in which A and B denote two different kinds of tag structures, two C -repeated patterns will be found: AB and BA . If only one C -repeated pattern is found in a query result section, we assume that each repeat corresponds to a record. If multiple C -repeated patterns are found in a query result section, we need to select one of them to denote the record. The following heuristic is

used for the C-repeated pattern selection: The visual gap between two records in a query result section should be no smaller than any gap within a data record [Zhai and Liu 2006]. We select the C-repeated pattern that satisfies this constraint.

4.4 Data Value Alignment and Label Assignment

As indicated by the third block in Component 2 of Figure 3, data value alignment and label assignment can be performed simultaneously as a single step; that is, for each data value, ODE first assigns a label to it. Next, the data values with the same label are aligned to the same column in the final table and the shared label assigned as the column's label.

Given a data value d , its history h is composed of the features that occur for d . The probability of a history h together with an attribute a is defined as [Ratnaparkhi 1996]

$$p(h, a) = \pi \prod_{j=1}^k \lambda_j^{f_j(h, a)}, \quad (8)$$

where π is a normalization constant and λ_j , which is described in Section 4.1, is the positive parameter corresponding to feature f_j . Hence, the probability that d is generated from an attribute a is

$$P(a|h) = \frac{p(h, a)}{\sum_{a \in A} p(h, a')}, \quad (9)$$

where a' refers to any attribute in the attribute set A .

Given a sequence of raw data strings $\{d_1, \dots, d_m\}$, an attribute sequence candidate $\{a_1, \dots, a_m\}$ has conditional probability

$$P(a_1, \dots, a_m | d_1, \dots, d_m) = \prod_{i=1}^m p(a_i | h_i). \quad (10)$$

Subsequently, label assignment can be performed by finding a most probable attribute for a data value from the potential attributes within its context.

The label assignment procedure searches over the candidate labels for the data values in the QRR and the label sequence that has the largest probability is chosen to be the correct set of labels. A k-beam search algorithm is used to find the label sequence with the largest probability given a QRR [Feiner et al. 2003].

Let $D = \{d_1, \dots, d_n\}$ be a QRR in which d_i refers to the i^{th} data value in the QRR and s_{ij} is the j^{th} largest probability label sequence up to and including d_i . The algorithm shown in Figure 8 is used to find the label sequence with the largest probability. A k-beam search algorithm is used to avoid exponential complexity if we exhaustively search for the label sequence with the largest probability. Given the first data value d_1 , we set k label sequences s_{1j} , $1 \leq j \leq k$ as the top k labels with the largest probability for d_1 (line 1). From the second data value d_2 to the last data value, each label sequence $s_{(i-1)j}$ is extended to multiple label sequences s_{ij} to make a new sequence by appending new labels onto $s_{(i-1)j}$. A new set of k label sequences is selected from the set of newly generated label sequences (lines 2–8). The final result is that all the data values are labeled with the label sequence with the largest probability.

Procedure labelAssignment()

1. Assign labels for d_1 , find top k labels with largest probability, set s_{1j} , $1 \leq j \leq k$ accordingly.
2. **for** $i = 2$ **to** n
3. **for** $j = 1$ **to** k
4. Assign labels for d_i , given $s_{(i-1)j}$ as previous label sequence, and append each label to s_{ij} to make a new sequence.
5. **endfor**
6. Find k label sequences with the largest probability from the newly generated label sequence and set s_{ij} , $1 \leq j \leq k$, correspondingly.
7. **endfor**
8. Label the data value using the label sequence s_{n1} , which has the largest probability.

Fig. 8. Largest probability label sequence searching algorithm.

Table IV. Summary of Datasets

Domain	# web sites	# MRP	# SRP	# ERP
Book	40	290	43	67
Airfare	40	317	36	47
Music	40	294	53	53
Movie	40	346	48	6
<i>Total</i>	160	1247	180	173

5. EXPERIMENTS

In this section, we present experiments to show the effectiveness of ODE. The experiments are run on query result pages collected from the Web sites in four popular ecommerce domains: Book, Airfare, Music, and Movie. We compare the experimental results of ODE with the primary wrapper PADE and with another state-of-the-art wrapper DeLa [Wang and Lochovsky 2003].

5.1 Datasets

Table IV summarizes the datasets used in our experiments. For each of the four domains, 40 ecommerce Web sites are explored and 10 queries are prepared. For each Web site, each of the 10 queries in the corresponding domain is submitted and the query result pages are collected. If there is more than one page returned for a given query, only the first page is collected. We manually inspected the collected query result pages and divided them into three classes according to the number of QRRs they contain.

- (1) *Multiple Result Page (MRP)*: A page that contains two or more than two query result records.
- (2) *Empty Result Page (ERP)*: A page that contains no query result. An empty result page may not be available for some Web sites.
- (3) *Single Result Page (SRP)*: A page that contains only one query result. A single result page may not be available for some Web sites.

In the 1,247 MRPs, there are 301 MRPs in which the QRRs are displayed in at least two sections. We also found that many Web sites return some other records if there are actually no or few records that fully match the query.

Table V. Data Extraction Accuracies for Columns in Training and New Web Sites

Domain	<i>Training Web Sites</i>		<i>New Web Sites</i>	
	Precision	Recall	Precision	Recall
Book	86.2%	83.4%	85.6%	82.5%
Airfare	89.6%	89.2%	89.5%	89.1%
Music	88.1%	83.3%	86.8%	83.1%
Movie	86.7%	83.5%	86.0%	82.6%
<i>Average</i>	87.6%	84.8%	87.0%	84.3%

In our experiment, we first use DeLa to extract and label the data to reduce the workload. A double-check is performed to inspect the output of DeLa to generate the comparison baseline.

5.2 Experiment Results

The Web sites are divided into two categories according to whether they are used to construct the domain ontology.

- (1) *Training Web sites* are Web sites that are used to construct the domain ontology.
- (2) *New Web sites* are Web sites that have not been used to construct the domain ontology.

We ran a four-fold cross-validation on the collected dataset to take full advantage of the data; that is, for each round, 30 out of the 40 Web sites for each domain are used as training Web sites to derive the domain ontology and the remaining 10 Web sites are new Web sites. Since PADE and DeLa do not require training Web sites, we do not classify the Web sites into the two categories for them in our experiments.

5.2.1 Data Extraction Accuracy. Table V shows the data extraction accuracy for the columns in the training and new Web sites. The precision is the ratio of the correctly identified, aligned, and labeled columns over all labeled columns and the recall is the ratio of the correctly identified, aligned, and labeled columns over all columns. It can be seen that ODE correctly identifies, aligns, and labels columns with precision around 87% and recall around 84%, in both training and new Web sites.

5.2.2 Query Result Section Identification Accuracy. In our experiment, we assume that a query result section is correctly identified if all and only QRR information is included in the query result section. Table VI shows the query result section identification accuracy for the training Web sites. It can be seen that ODE has excellent performance for any type of query result page. Similar accuracy can be obtained for query result section identification on new Web sites, as shown in Table VII. Table VIII shows the query result section identification accuracy of the three wrappers: ODE, PADE, and DeLa. It can be seen that ODE outperforms both PADE and DeLa.

Table VI. Query Result Section Identification Accuracy for Training Web Sites

Domain	MRP	SRP	ERP
Book	97.9%	93.0%	95.5%
Airfare	100%	97.2%	95.8%
Music	98.0%	96.2%	94.3%
Movie	97.7%	95.8%	100%
<i>Average</i>	98.4%	95.6%	95.4%

Table VII. Query Result Section Identification Accuracy for New Web Sites

Domain	MRP	SRP	ERP
Book	97.6%	95.3%	95.5%
Airfare	100.0%	91.7%	95.7%
Music	97.3%	98.1%	96.2%
Movie	97.7%	93.8%	100.0%
<i>Average</i>	98.2%	95.0%	96.0%

Table VIII. Query Result Section Identification Accuracy of the Three Wrappers

Page type	ODE		PADE	DeLa
	Training	Test		
MRP	98.4%	98.2%	97.3%	92.6%
SRP	95.6%	95.0%	0%	86.7%
ERP	95.4%	96.0%	0%	88.4%
<i>Average</i>	97.8%	97.6%	75.8%	91.5%

Discussion. In our experiment, ODE identified the query result section effectively for a Multiple Result Page (MRP) except for the case when the query result records are arranged into two or more different formats in which only one format will be identified as the query result section according to its area, distance to the center of the HTML page, and the number of data values in each query result record.

For a single result page (SRP), most errors result from the following problems:

- (1) Identifying a subtree of the page's tag tree that is larger than the minimal subtree that contains the query result section because the additional labels were incorrectly identified to be relevant to the ontology.
- (2) Identifying a subtree of the page's tag tree that is smaller than the minimal subtree that contains the query result section because the additional labels were incorrectly identified to be irrelevant to the ontology.

Some empty result pages (ERPs) are incorrectly identified as MRPs or SRPs because these pages contain advertisement information, which may be identified as query result records since they have high correlation with the ontology.

5.2.3 Data Value Alignment Accuracy. To evaluate the accuracy of the data value alignment process (see Section 4.4), we use two sets of metrics due to the possibly wide variation in the number of attributes in a query result page.

The first set of metrics calculates the precision⁹ defined over the attribute level. Given an attribute a and the data values it generates, the alignment accuracy is

$$P_a = \frac{C_c}{C_a}, \quad (11)$$

where C_c is the count of correctly aligned data values and C_a is the count of data values generated by the attribute a . It should be noted that the alignment accuracy is defined according to the QRR alignment consistency. An alignment is accurate if the data values generated by an attribute are aligned into the same column. In the experiment, if the data values generated by one attribute are not aligned into the same column, we select the column that contains the most number of data values generated by this attribute as the correctly aligned column, and mark those data values in this column to be the correctly aligned values. Thus, the rest of the values that are not aligned to this column are marked as incorrectly aligned. For example, suppose we have extracted five data values d_1, d_2, d_3, d_4 , and d_5 that are generated by an attribute a and they should be aligned into the same column. After the data value alignment process, the data values in d_1, d_2 , and d_3 are aligned into one column and those in d_4 and d_5 are aligned into another column. Consequently, we assume that d_1, d_2 , and d_3 are the correctly aligned data values (i.e., $C_c = 3$) and the accuracy for the attribute a is $3/5 = 60\%$. Given a page, its attribute-level accuracy P_r is defined to be the average of all its attributes' accuracy.

We further use the following page-level precision to evaluate the performance of the alignment method. We have

$$P_p = \frac{C_p}{N_a}, \quad (12)$$

in which C_p is the count of pages in which the data values in all QRRs are correctly aligned and N_a is the count of pages from which we need to extract QRRs. It can be seen that the page-level metric is stricter than the record-level metric because it assumes that an incorrectly aligned record in a page makes the page fail (i.e., all the records in the page fail), which causes P_p to be smaller than P_r in general.

Table IX and Table X show the data value alignment accuracy for Multiple Record Pages (MRPs) in the training Web sites and the new Web sites.¹⁰ The alignment accuracy of records from the training and new Web sites is similar. Both accuracies are around 99% at the attribute level and the accuracy is around 97% at the page level. Table XI shows the data value alignment accuracy of the three wrappers ODE, PADE, and DeLa. It can be seen that while all three wrappers achieve very high accuracy for data value alignment, ODE still slightly outperforms the other two wrappers.

⁹Note that we do not need to measure recall here, since each data value is aligned to one of the columns (correctly or not).

¹⁰Table IX and Table X show the data value alignment accuracy only for Multiple Record Pages (MRPs) because Single Record Pages (SRPs) and Empty Record Pages (ERPs) do not need the data value alignment step.

Table IX. Data Value Alignment Accuracy for Multiple Record Pages (MRPs) in Training Web Sites

Domain	P_r	P_p
Book	99.6%	96.5%
Airfare	99.6%	98.5%
Music	98.9%	97.2%
Movie	99.2%	96.1%
<i>Average</i>	99.3%	97.1%

Table X. Data Value Alignment Accuracy for Multiple Record Pages (MRPs) in New Web Sites

Domain	P_r	P_p
Book	99.2%	96.1%
Airfare	99.3%	98.6%
Music	98.6%	97.4%
Movie	98.8%	96.0%
<i>Average</i>	99.0%	97.0%

Table XI. Data Value Alignment Accuracy of the Three Wrappers

Metric	ODE	PADE	DeLa
P_r	99.3%	98.9%	98.4%
P_p	97.1%	96.7%	95.1%

5.2.4 Label Assignment Accuracy. Table XII shows the label assignment accuracy for the correctly aligned columns in the training and new Web sites. The precision is the ratio of the correctly labeled columns over all labeled columns and the recall is the ratio of the correctly labeled columns over all columns. Recall is usually lower than precision because we do not label columns whose labels never appear in any query interfaces or query result pages. It can be seen that ODE has a label assignment precision and recall around 90% and 87% for columns in both training and new Web sites, both of which are much higher than those of DeLa.

Discussion. The reason that the label recall of ODE is much higher than that of DeLa is that DeLa labels columns with labels only from the query result page or query interface from the same Web site. Consequently, many columns cannot be labeled. On the other hand, ODE may label columns with labels from other Web sites in the same domain. The reason that the label precision of ODE is higher than that of DeLa is that ODE uses more domain knowledge, such as the label sequence and the occurrence probability. This additional knowledge helps to assign a correct label for a column. For ODE, around 4% of columns are not labeled in the aligned table, which usually means that no suitable labels are found in the query interfaces or query result pages from the domain.

5.2.5 Effect of the Number of Training Web Sites. Figure 9 shows the record-level precision and recall for data value alignment and the label

Table XII. *QRR* Label Assignment Accuracies for Columns in Training and New Web Sites

Domain	ODE					
	<i>Training Web Sites</i>		<i>New Web Sites</i>		DeLa	
	Precision	Recall	Precision	Recall	Precision	Recall
Book	89.2%	86.2%	88.9%	86.0%	81.2%	73.3%
Airfare	92.4%	92.4%	92.7%	91.7%	86.4%	84.6%
Music	90.7%	86.3%	89.9%	86.4%	80.3%	73.7%
Movie	89.8%	86.6%	89.2%	86.4%	74.8%	70.8%
Average	90.5%	87.1%	90.1%	86.8%	80.2%	75.5%

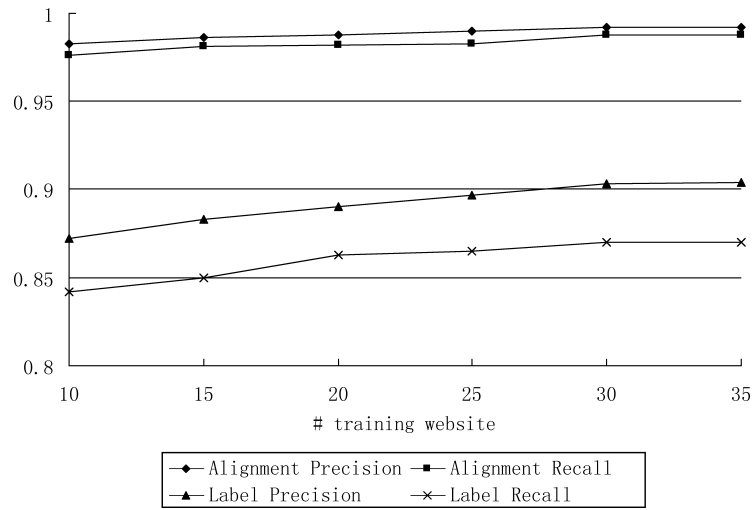


Fig. 9. Effect of number of training Web sites.

assignment precision and recall as the number of training Web sites increases. It can be seen that all the four metrics increase steadily when more training Web sites are available, although the effect is different for data value alignment and label assignment. On the one hand, data value alignment already has high precision and recall when the number of training Web sites is 10 and the precision increase is marginal as the number of training Web sites increases. On the other hand, the precision and recall of label assignment initially increase rapidly as the number of training Web sites increases to 30, because more labels are available, but is fairly stable thereafter because the number of unique labels does not increase much after about 30 training Web sites.

6. RELATED WORK

In recent years, the volume and quality of deep Web information has attracted much research attention. As the returned data for a query are embedded in HTML pages, much research has focused on extracting the data from these query result pages. Simultaneously, many researchers have studied the

problem of extracting information from HTML files. Earlier works focused on wrapper induction, during which human assistance is required to build the wrapper. Recently, several data extraction methods have been proposed to automatically extract the records from a query result page.

6.1 Wrapper-Based Data Extraction

In early work on wrapper induction, extraction rules are semi-automatically derived based on inductive learning. A user labels or marks the data to extract (the target data) in a set of training pages or a list of data records in a page, and the system then learns the extraction rules from the labeled data and uses them to extract records from new pages. A rule usually contains two patterns, a prefix pattern and a suffix pattern, to denote the beginning and the end, respectively, of the target data. Some existing wrapper induction systems include WL² [Cohen et al. 2002], SoftMealy [Hsu and Dung 1998], WIEN [Kushmerick 2000], and Stalker [Muslea et al. 1999].

Semi-automatic wrapper induction has the advantage that no extraneous data are extracted as the user can label only the data in which he/she is interested. Furthermore, these methods are usually very fast at extracting data from Web pages, faster than most other kinds of techniques, including ODE. Hence, many real-time applications, such as metasearch, use wrapper induction techniques to extract data from Web pages.

However, semi-automatic wrapper induction requires labor-intensive and time-consuming manual labeling of data. Hence, it is not scalable to a large number of Web sites. Moreover, an existing wrapper usually performs poorly when the format of a query result page changes. Considering that the Web changes rapidly, the format of a query result page may change frequently. Hence, wrapper induction involves two further difficult problems: page monitoring to determine whether a page's format has changed and wrapper maintenance to maintain a wrapper when a page's format changes.

To overcome the problems of semi-automatic wrapper induction, some unsupervised learning methods, such as RoadRunner [Crescenzi et al. 2001], Omini [Buttler et al. 2001], ExAlg [Arasu and Garcia-Molina 2003], IEPAD [Chang and Lui 2001], DeLa [Wang and Lochovsky 2003], and PickUp [Chen et al. 2004], have been proposed to fully automatically extract the data from a query result page based on the tag structure that exists in one or several HTML pages from the same Web site.

RoadRunner [Crescenzi et al. 2001] starts with any page as its initial page template and then compares this template with each new page. If the template cannot generate the new page, it is fine-tuned. However, RoadRunner suffers from several limitations.

- (1) When RoadRunner finds that the current template cannot generate a new page, it searches through an exponential size page schema trying to fine-tune the template.
- (2) RoadRunner assumes that the template generates all HTML tags, which does not hold for many Web databases.

- (3) RoadRunner assumes that there are no disjunctive attributes, which the authors of RoadRunner admit does not hold for many query result pages.
- (4) Data labeling/annotation is not addressed in RoadRunner, since it mainly focuses on the data extraction problem.

Omini [Buttler et al. 2001] uses several heuristics to extract a subtree that contains data strings of interest from an HTML page. Then, another set of heuristics is used to find a separator to segment the minimum data object-rich¹¹ subtree into data records. However, the results reported in Liu et al. [2003] indicate that Omini has low effectiveness (recall 39% and precision 56%) for extracting data records.

ExAlg [Arasu and Garcia-Molina 2003] works on a set of Web pages from the same Web site and computes equivalence classes, which are sets of tokens having the same frequency of occurrence on all input pages. Large and frequently occurring equivalence classes are extracted for page template generation. ExAlg has several problems. First, it assumes that a tag or string is used to separate data strings, which is not valid for many Web sites. Second, while, ExAlg can handle optional and disjunctive attributes, it cannot handle pages that contain lists. Finally, it is possible that multiple equivalence classes are extracted, which would then require human involvement to select one.

IEPAD [Chang and Lui 2001] first encodes all HTML tokens of a parsed Web page into a binary sequence and then uses a PAT tree and heuristics to find frequent patterns. The users can choose one of the generalized patterns as an extraction rule. While IEPAD is efficient for Web pages that only contain plain-structured data, the approach is not fully automatic and cannot handle nested-structured data with multivalued attributes.

DeLa [Wang and Lochovsky 2003] models the data strings contained in template-generated Web pages as string instances, encoded in HTML tags, of the implied nested type of their Web database. A regular expression is employed to model the HTML-encoded version of the nested type. Since the HTML tag structure enclosing a data string may appear repeatedly if the page contains more than one instance of the data string, the page is first transformed into a token sequence composed of HTML tags and a special token “text” representing any text string enclosed by pairs of HTML tags. Then, C-repeated substrings are extracted from the token sequence and a regular expression wrapper is induced from the repeated substrings according to some hierarchical relationships among them. The main problem with this method is that it often produces multiple patterns (rules) and it is hard to decide which is correct. PickUp [Chen et al. 2004] identifies table structures in Web pages by also mining repeated patterns in HTML tag sequences.

TISP [Tao and Embley 2007] constructs wrappers by looking for commonalities and variations in sibling tables in sibling pages (i.e., pages that are from the same Web site and have similar structures). Commonalities in sibling tables represent labels, while variations represent data values. Matching sibling

¹¹An object-rich subtree corresponds to what we refer to as the query result section of a query result page.

tables are found using a tree-mapping algorithm applied to the DOM tree representation of tagged tables in sibling pages. Using several predefined table structure templates, which are described using regular expressions, TISP “fits” the tables into one of the templates allowing the table to be interpreted. TISP is able to handle nested tables as well as variations in table structure (i.e., optional columns) and is able to adjust the predefined templates to account for various combinations of table templates. The whole process is fully automatic and experiments show that TISP achieves an overall F-measure of 94.5% on the experimental dataset. However, TISP can only extract data records embedded in HTML `<table>` tags and only works when sibling pages and tables are available.

Several factors make it very difficult to derive accurate wrappers based solely on HTML tags [Zhao et al. 2005]. First, since HTML tags are often used in unexpected and unconventional ways, we cannot rely on “proper” HTML tag usage. Second, since the main purpose of HTML tags is to facilitate the rendering of Web pages, they convey little semantic information about the data strings. Consequently, an ill-structured HTML page may still display correctly. Furthermore, some data strings may contain embedded tags, which may confuse the wrapper generators, making them even less reliable.

Recently, to overcome these shortcomings, visual features have been used for data extraction. In DEPTA [Zhai and Liu 2006], the intuition that the gap within a QRR is typically smaller than that between QRRs is used to segment data records and to identify individual data records. The data strings in the records are then aligned, for those data strings that can be aligned with certainty, based on tree matching. In ViPER [Simon and Lausen 2005] and ViNTs [Zhao et al. 2005], both visual content features and the HTML tag structure are used to segment the data records. DEPTA, ViNTs, and ViPER can all handle nested data structures. However, ViNTs requires a no-query-result page and multiple QRRs (at least four) in each nonempty query result page to generate an accurate wrapper and none of them does label assignment.

6.2 Ontology-assisted Data Extraction

In Embley et al. [1999], an ontology-assisted data extraction method is proposed to extract the records from query result pages using an ontology constructed by an expert. The ontology (i.e., a conceptual model instance) describes the data of interest, including relationships, lexical appearance, and context keywords. By parsing the ontology, a database schema and recognizers for constants and keywords are produced and then are invoked to recognize and extract data from unstructured documents and structure it according to the generated database schema. However, the approach used by Embley et al. [1999] to identify the query result section may not be realistic, as it simply searches for the subtree in the HTML tag tree with the largest fan-out.

Snoussi et al. [2001] propose an ontology-assisted method to extract the data in three steps: (i) convert an HTML page into XML; (ii) construct a data model using the ontology; and (iii) map the XML document to the elements in the ontology. The data strings are converted to a predefined format so that other software can use them.

Creating an ontology is difficult, time consuming, and usually requires many experts to cooperate. To solve these problems, some semi-automatic or automatic methods are proposed recently to construct an ontology from databases or tables. Vivan and Heuser [2002] present a semi-automatic process to generate an ontology from relational databases in two parts: First, reverse engineer from the relational database schema to the ontology and then generate regular expressions from the database data instances. The process is semi-automatic because user decisions are required for alternatives of ontology generation.

Roitman and Gal [2006] developed OntoBuilder to automatically extract ontologies from Web query interfaces and then used them to generate the global domain ontology. OntoBuilder showed that the precedence order of multiple attributes helps improve the ontology matching result. In Tijerino et al. [2005], an approach, called TANGO, tries to automatically generate an ontology from the tables in the query result pages in four consecutive steps: (i) recognize and normalize table information; (ii) construct mini-ontologies from the normalized tables; (iii) discover interontology mappings; and (iv) merge mini-ontologies into a growing application ontology. The difference between the ontology construction approaches used by OntoBuilder, TANGO, and ODE is that OntoBuilder and TANGO consider domain information contained either in query interfaces (OntoBuilder) or query result pages (TANGO), while ODE considers the domain information contained in both query interfaces and query result pages.

DeepMiner learns a domain ontology for semantically marking up Web services [Wu et al. 2005]. DeepMiner collects the domain knowledge from the source Web sites' query interface and data pages. The ontology of DeepMiner contains similar components as the ODE ontology, such as *synonyms* in DeepMiner versus *alias* in ODE, *instance of concept* in DeepMiner versus *value* in ODE, etc. Compared with ODE, DeepMiner's goal is simply to build the ontology for the source Web services, while ODE further addresses the problem of how to utilize the ontology to extract and annotate data embedded in query result pages.

6.3 Label Assignment

Recent work on label assignment includes DeLa [Wang and Lochovsky 2003], TISP [Tao and Embley 2007], and TISP++ [Tao and Embley 2009] and the work of Lu et al. [2007]. DeLa uses four heuristics described in Section 3.3, from which our method starts, to label the query result table [Wang and Lochovsky 2003].

TISP [Tao and Embley 2007] uses the labels that its finds (i.e., the commonalities shared by sibling tables) when constructing a wrapper for a Web page to annotate the data extraction result. TISP++ [Tao and Embley 2009] further augments TISP by generating an OWL¹² ontology for a Web site. For each table label, TISP++ generates an OWL class. The label name becomes the class name. If a label is paired with an actual value, TISP++ generates an OWL data type

¹²The Web Ontology Language (OWL) is a knowledge representation language endorsed by W3C for authoring ontologies.

property for the OWL class associated with this label. After the OWL ontology is generated, TISP++ automatically annotates the pages from this Web site with respect to the generated ontology and outputs it as an RDF file suitable for querying using SPARQL. The limitation of TISP++ is that it only generates an OWL ontology for a single set of sibling pages from the same Web site and does not combine ontologies generated from different Web sites in a domain. If the generated ontology is unable to capture all the data semantics of the site (e.g., when voluntary labels are not available in the Web pages), then when it is applied to annotate the rest of the pages from the same Web site, there may still be some data that cannot be labeled. In contrast, ODE tries to generate the ontology for multiple Web sites in a domain (i.e., a domain ontology) by utilizing the complementary information from the different Web sites.

The method in Lu et al. [2007] aggregates several annotators, most of which are based on the same heuristics as used in Wang and Lochovsky [2003]. One of the unique proposed annotators, *schema value annotator*, employs an integrated interface schema and tries to match the data values with attributes in the “global” schema. Once a match is discovered, the attribute name from the global schema can be used to label the matched data values. To our knowledge, the work by Lu et al. [2007] is the most recent and relevant work to ODE. Both ODE and Lu et al. [2007] share some common heuristics in data annotation including the idea of matching data values to a global representation of domain knowledge. In Lu et al.’s work, such global domain knowledge is represented in an integrated interface schema, while in ODE it is represented in a domain ontology. The difference between Lu et al.’s work and ODE lies in the algorithm for combining multiple annotation heuristics/features. Lu et al. [2007] assume independence between different annotation features and consider each of them individually. When a conflict arises because two or more features generate different labels, the label with the highest *success rate* in the past will be chosen. Furthermore, Lu et al.’s work also assumes a *context-free* annotation process, that is, the previous data values’ assigned labels do not affect how to label the subsequent data values. By contrast, ODE considers multiple features when calculating the probability of assigning a label to a target data value. In addition, ODE also takes the labeling sequence into account (i.e., the previous labels assigned could affect the next label to assign).

7. CONCLUSIONS AND FUTURE WORK

In this article, a novel Ontology-assisted Data Extraction method, ODE, is introduced. In ODE, the ontology for a domain is constructed by matching the query interfaces and the query result pages among different Web sites. Then, the ontology is used to do the data extraction. For query result section identification, ODE finds a subtree, which has the maximum correlation with the ontology, in the HTML tag tree. For data value alignment and label assignment, ODE uses a maximum entropy model. Context, tag structure, and visual information are used as features for the maximum entropy model. The ontology-assisted data extraction method is fully automatic and can avoid many of the problems that exist in most current automatic data extraction methods. Experimental results

show that ODE is very effective and can satisfactorily address most existing shortcomings of data extraction approaches.

Despite the effectiveness of ODE, there is still much that can be improved. One limitation of ODE is that to label attributes it is necessary that the labels appear in the query interfaces or query result pages within a domain. However, there are some attributes whose labels never appear in any query interface or query result page. Consequently, such attributes cannot be labeled. To overcome this limitation, it might be possible to use the Web, as a repository of knowledge, to assign a suitable label for these kinds of attributes. By sending the attribute value and appropriate domain information of such attributes to a search engine, it may be possible to analyze the returned pages, using natural language processing techniques, to discover suitable labels for the attributes. An already mentioned limitation of ODE is that if the query result records are arranged into two or more different formats in the query result pages, then only one format will be identified as the query result section. For such query result pages, it may be possible to use information in the generated ontology to identify the multiple query result sections. Finally, the performance of ODE on certain types of query result pages is far from satisfactory. As one example, for some pages with factor values,¹³ the factor values may sometimes be removed. As another example, for pages with off-page links to a significant amount of desired data, ODE does not extract those desired data. Techniques that could overcome these limitations, such as that used in Lerman et al. [2004], require further exploration.

REFERENCES

- ARASU, A. AND GARCIA-MOLINA, H. 2003. Extracting structured data from Web pages. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 337–348.
- BAEZA-YATES, R. 1989. Algorithms for string matching: A survey. *ACM SIGIR Forum* 23, 3-4, 34–58.
- BERGER, A. L., DELLA-PIETRA, S. A., AND DELLA-PIETRA, V. J. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.* 22, 1, 39–71.
- BERGMAN, M. K. 2001. The deep Web: Surfacing hidden value. White paper, BrightPlanet Corporation. <http://www.brightplanet.com/resources/details/deepweb.html>.
- BILKE, A. AND NAUMANN, F. 2005. Schema matching using duplicates. In *Proceedings of the 21st IEEE International Conference on Data Engineering*. 69–80.
- BORTHWICK, A. 1999. A maximum entropy approach to named entity recognition. Ph.D. thesis, Computer Science Department, New York University.
- BUTTLER, D., LIU, L., AND PU, C. 2001. A fully automated object extraction system for the World Wide Web. In *Proceedings of the 21st International Conference on Distributed Computing Systems*. 361–370.
- CHANG, C. H. AND LUI, S. C. 2001. IEPAD: Information extraction based on pattern discovery. In *Proceedings of the 10th World Wide Web Conference*. 681–688.
- CHANG, K. C.-C., HE, B., LI, C., PATEL, M., AND ZHANG, Z. 2004. Structured databases on the Web: Observations and implications. *SIGMOD Rec.* 33, 3, 61–70.
- CHEN, L., JAMIL, H. M., AND WANG, N. 2004. Automatic composite wrapper generation for semi-structured biological data based on table structure identification. *SIGMOD Rec.* 33, 2, 58–64.

¹³Factored values are those values that appear once for a group of records (e.g., year value 2008 appears once for the group of 2008 cars, year value 2007 appears once for the group of 2007 cars, etc.).

- COHEN, W., HURST, M., AND JENSEN, L. 2002. A flexible learning system for wrapping tables and lists in HTML documents. In *Proceedings of the 11th World Wide Web Conference*. 232–241.
- CRESCENZI, V., MECCA, G., AND MERIALDO, P. 2001. Roadrunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*. 109–118.
- EMBLEY, D. W., CAMPBELL, D. M., JIANG, Y. S., LIDDLE, S. W., LONSDALE, D. W., NG, Y.-K., AND SMITH, R. D. 1999. Conceptual-model-based data extraction from multiple-record Web pages. *IEEE Trans. Data Knowl. Engin.* 31, 3, 227–251.
- FEINER, A., KRAUS, S., AND KORE, R. E. 2003. KBFS: K-best-first search. *Ann. Math. Artif. Intell.* 39, 1-2, 19–39.
- GRAVANO, L., IPEIROTIS, P. G., AND SAHAMI, M. 2003. QProber: A system for automatic classification of hidden Web databases. *ACM Trans. Inform. Syst.* 21, 1, 1–41.
- GUSFIELD, D. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK.
- HE, B. AND CHANG, K. C.-C. 2006. Automatic complex schema matching across Web query interfaces: A correlation mining approach. *ACM Trans. Datab. Syst.* 31, 1, 346–396.
- HSU, C.-N. AND DUNG, M.-T. 1998. Generating finite-state transducers for semi-structured data extraction from the Web. *Inform. Syst.* 23, 8, 521–538.
- KUSHMERICK, N. 2000. Wrapper induction: Efficiency and expressiveness. *Artif. Intell.* 118, 1–2, 15–68.
- LERMAN, K., GETOOR, L., MINTON, S., AND KNOBLOCK, C. 2004. Using the structure of Web sites for automatic segmentation of tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 119–130.
- LIU, B., GROSSMAN, R., AND ZHAI, Y. 2003. Mining data records in Web pages. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 601–606.
- LU, Y., HE, H., ZHAO, H., MENG, W., AND YU, C. 2007. Annotating structured data of the deep Web. In *Proceedings of the 23rd IEEE International Conference on Data Engineering*. 376–385.
- MINKA, T. 2003. A comparison of numerical optimizers for logistic regression. Tech. rep., Department of Statistics, Carnegie Mellon University.
- MUSLEA, I., MINTON, S., AND KNOBLOCK, C. 1999. A hierarchical approach to wrapper induction. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*. 190–197.
- RATNAPARKHI, A. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the 1st Empirical Methods in Natural Language Processing Conference*. 133–141.
- ROITMAN, H. AND GAL, A. 2006. Ontobuilder: Fully automatic extraction and consolidation of ontologies from Web sources using sequence semantics. In *Proceedings of the EDBT Workshops*. 573–576.
- SIMON, K. AND LAUSEN, G. 2005. ViPER: Augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. 381–388.
- SNOUSSI, H., MAGNIN, L., AND NIE, J.-Y. 2001. Heterogeneous Web data extraction using ontologies. In *Proceedings of the Conference on Agent-Oriented Information Systems*. 99–110.
- SU, W., WANG, J., AND LOCHOVSKY, F. H. 2006. Holistic schema matching for Web query interfaces. In *Proceedings of the 10th International Conference on Extending Database Technology*. 77–94.
- SU, W., WANG, J., LOCHOVSKY, F. H., AND LIU, Y. 2009. PADE: Pair-wise alignment-based data extraction. Tech. rep. HKUST-CS09-01, *Department of Computer Science and Engineering*, The Hong Kong University of Science and Technology, Hong Kong.
- TAO, C. AND EMBLEY, D. W. 2009. Automatic hidden-Web table interpretation, conceptualization, and semantic annotation. *Data Knowl. Engin.* 68, 7, 683–703.
- TAO, C. AND EMBLEY, D. W. 2007. Automatic hidden-Web table interpretation by sibling page comparison. In *Conceptual Modeling – ER’07. Lecture Notes in Computer Science*, vol. 4801 Springer Berlin, 566–581.
- TIJERINO, Y. A., EMBLEY, D. W., LONSDALE, D. W., DING, Y., AND NAGY, G. 2005. Towards ontology generation from tables. *World Wide Web* 8, 3, 261–285.
- VIVAN, O. M. AND HEUSER, C. A. 2002. Semiautomatic generation of data-extraction ontologies from relational databases. In *Proceedings of the XVII Simpósio Brasileiro de Banco de Dados*. 252–262.

- WANG, J. AND LOCHOVSKY, F. H. 2003. Data extraction and label assignment for Web databases. In *Proceedings of the 12th World Wide Web Conference*. 187–196.
- WANG, J., WEN, J., LOCHOVSKY, F. H., AND MA, W. Y. 2004. Instance-Based schema matching for Web databases by domain-specific query probing. In *Proceedings of the 30th International Conference on Very Large Data Bases*. 408–419.
- WORLD WIDE WEB CONSORTIUM. 1999. HTML 4.01 specification.
<http://www.w3.org/TR/REC-html40/>.
- WU, W., DOAN, A., YU, C., AND MENG, W. 2005. Boot-strapping domain ontology for semantic Web services from source Web sites. In *Proceedings of the 6th VLDB Workshop on Technologies for E-Services*. 11–12.
- ZHAI, Y. AND LIU, B. 2006. Structured data extraction from the Web based on partial tree alignment. *IEEE Trans. Knowledge Data Engin.* 18, 12, 1614–1628.
- ZHANG, T. AND OLES, F. J. 2001. Text categorization based on regularized linear classification methods. *Inform. Retrieval*. 4, 1, 5–31.
- ZHAO, H., MENG, W., WU, Z., RAGHAVAN, V., AND YU, C. 2005. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th World Wide Web Conference*. 66–75.

Received February 2008; revised January 2009; accepted February 2009