Extracting Data Records from the Web Using Tag Path Clustering

Gengxin Miao[1] * Junichi Tatemura[2] Wang-Pin Hsiung[2] Arsany Sawires[2] Louise E. Moser[1]
[1] ECE Dept., University of California, Santa Barbara, Santa Barbara, CA, 93106
[2] NEC Laboratories America, 10080 N. Wolfe Rd SW3-350, Cupertino, CA, 95014
{miao,moser}@ece.ucsb.edu, {tatemura,whsiung,arsany}@sv.nec-labs.com

ABSTRACT

Fully automatic methods that extract lists of objects from the Web have been studied extensively. Record extraction, the first step of this object extraction process, identifies a set of Web page segments, each of which represents an individual object (*e.g.*, a product). State-of-the-art methods suffice for simple search, but they often fail to handle more complicated or noisy Web page structures due to a key limitation – their greedy manner of identifying a list of records through pairwise comparison (*i.e.*, similarity match) of consecutive segments. This paper introduces a new method for record extraction that captures a list of objects in a more robust way based on a holistic analysis of a Web page. The method focuses on how a distinct *tag path* appears repeatedly in the DOM tree of the Web document. Instead of comparing a pair of individual segments, it compares a pair of tag path occurrence patterns (called *visual signals*) to estimate how likely these two tag paths represent the same list of objects. The paper introduces a similarity measure that captures how closely the visual signals appear and interleave. Clustering of tag paths is then performed based on this similarity measure, and sets of tag paths that form the structure of data records are extracted. Experiments show that this method achieves higher accuracy than previous methods.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Data mining; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithm, Performance, Experimentation

Keywords

Information extraction, data record extraction, clustering

1. INTRODUCTION

The Web contains a large amount of structured data, and serves as a good user interface for databases available over the Internet. A large amount of Web content is generated from databases in response to user queries. Such content is sometimes referred to as the *deep Web*. A deep Web page typically displays search results as a list of objects (*e.g.*, products) in the form of structured data rendered in HTML. A study in 2004 found 450,000 databases in the deep Web [5]. Structured data also plays a significant role on the *surface Web*. Google estimated that their crawled data set contains 154 million *Web tables*, *i.e.*, relational data rendered as HTML tables [3]. In addition to relational tables, the Web contains a variety of lists of objects, such as conference programs and comment lists in blogs. It is an important and challenging task to identify such object lists embedded in Web pages in a scalable manner, which enables not only better search engines but also various applications related to Web data integration (*i.e.*, data mashups) and Web data mining (*e.g.*, blog analysis).

There have been extensive studies of fully automatic methods to extract lists of objects from the Web [1, 7]. A typical process to extract objects from a Web page consists of three steps: record extraction, attribute alignment, and attribute labeling. Given a Web page, the first step is to identify a *Web record* [11], *i.e.*, a set of HTML regions, each of which represents an individual object (*e.g.,* a product). The second step is to extract object attributes (*e.g.*, product names, prices, and images) from a set of Web records. Corresponding attributes in different Web records are aligned, resulting in spreadsheet-like data [19, 21]. The final step is the optional task (which is very difficult in general) of interpreting aligned attributes and assigning appropriate labels [17, 22].

In this paper we focus on Web record extraction. Our study is motivated by our experience in developing an automatic data extraction component of a data mashup system [16], where we scrape a set of objects from a *variety* of Web pages automatically. The extraction component, developed with existing state-of-the-art technologies, sometimes fails at the very first step, *i.e.*, record extraction, which significantly affects the entire mashup process.

Most state-of-the-art technologies for Web record extraction employ a particular similarity measure between Web page segments to identify a region in the page where a similar data object or record appears repeatedly. A representative example of this approach is MDR [11], which uses the edit distance between data segments (called *generalized nodes*). By traversing the DOM tree of a Web document, MDR discovers a set of consecutive sibling nodes that form a data region. More recent work [15, 19] extends this approach by introducing additional features such as the position of the rendered data. In our experience, an approach based on

*The work was performed during the author's internship at NEC.

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

MDR is sufficient for simple search, but it starts to fail as the Web page structure becomes more complicated.

We observe that, on many Web pages, objects are rendered in a highly decorated manner, which affects the quality of extraction. For instance, an image that is inserted between objects as a separator makes objects no longer consecutive. As a work around, we employ a heuristic rule to exclude decorative images from the DOM tree. In fact, such visual information can be helpful or harmful. A heuristic rule might utilize such decorations to identify object boundaries. However, it is not easy to generalize such a heuristic rule so that it applies to a variety of Web pages. Thus, in general, the irregularity that decorative elements introduce is more harmful than helpful. Moreover, as [21] notes, the same HTML tag can sometimes work as a template token (that contributes to form an object structure) and can sometimes work as a decorative element (that is used in an unstructured manner). Such tags can be very noisy but, if the algorithm ignores these tags, it can miss useful evidence of structured objects.

We also observe that objects are sometimes embedded in a complicated Web page structure with various context information. In such cases, objects are not necessarily rendered consecutively. Existing work tries to address such complex Web page structures [1]. However, that work typically assumes availability of multiple Web page instances.

A key limitation that we have identified in the MDR approach is its greedy manner of identifying a *data region* (a region containing records) through pairwise comparison of consecutive segments. In many cases, one misjudgment due to noise causes separation of an object list into multiple lists. We can imagine an extended algorithm that employs more sophisticated search for data regions instead of the greedy approach, but its computational cost is very high.

Thus, we propose an alternative approach to the Web record extraction problem, which captures a list of objects based on a holistic analysis of a Web page. Our method focuses on how a distinct *tag path* (*i.e.*, a path from the root to a leaf in the DOM tree) appears repeatedly in the document. Instead of comparing a pair of individual subtrees in the data, we compare a pair of tag path occurrence patterns (called *visual signals*) to estimate how likely these two tag paths represent the same list of objects. We introduce a similarity measure that captures how closely the tag paths appear and how they interleave. We apply clustering of tag paths based on this similarity measure, and extract sets of tag paths that form the structure of the data records.

Compared to existing approaches, our method has the following advantages:

- Data records do not have to be consecutive. Based on the discovery of non-consecutive data records, our method can also detect nested data records.
- Template tags and decorative tags are distinguished naturally. When a tag (path) appears randomly in unstructured content, the corresponding visual signal will not be similar to other signals. A tag (path) is clustered based on the structure of the data records only when it repeats similarly to other tags.

2. RELATED WORK

Extracting structured data from HTML pages has been studied extensively. Early work on wrapper induction utilizes manually labeled data to learn data extraction rules [9]. Such semi-automatic methods are not scalable enough for extraction of data on the scale of the Web. To address this limitation, more fully automatic methods have been studied recently. Fully automatic methods address two types of problems: (1) extraction of a set of objects (or data records) from a single page, and (2) extraction of underlying templates (or schema) from multiple pages [1, 7]. Our work focuses on the former, which does not assume the availability of multiple instance pages containing similar data records.

Techniques that address record extraction from a single page can be categorized into the following approaches, which evolved in this order: (a) early work based on heuristics [2], (b) mining repetitive patterns [4, 17], and (c) similarity-based extraction [11, 15, 21]. OMINI [2] applies a set of heuristics to discover separator tags between objects in a Web page, but is applicable to only simple cases. IEPAD [4] identifies substrings that appear multiple times in a document encoded as a token string. DeLa [17] extends that approach to support nested repetition, such as "(AB*C)*D". One limitation of such a pattern mining approach is that it is not robust against optional data inserted into records. The similarity-based approach tackles this limitation with approximate matching to identify repeating objects. MDR [11] is one such technique, which utilizes edit distance to assess whether two consecutive regions are a repetition of the same data type. It is reported that MDR out-performs both OMINI and IEPAD.

As discussed in Section 1, even similarity-based extraction has limitations when the data are complex and noisy. MDR relies on a greedy approach based on a similarity match between two segments, with a pre-determined threshold. A limitation of MDR is that it does not handle nested data objects. The researchers who developed MDR proposed an extended algorithm, NET, to address this issue [12]. NET handles nested objects by traversing a DOM tree in post-order (bottom-up), whereas MDR traverses the tree in pre-order (top-down). When a list of objects is discovered during traversal, the list is collapsed into a single object (pattern) so that the number of objects does not affect detection of higher-layer objects. However, NET still employs a greedy approach based on similarity match. Moreover, its bottom-up traversal with edit distance comparison is expensive. Whereas MDR's top-down traversal can stop as soon as it finds data records, NET's bottom-up traversal requires a full scan from the bottom up to the root. For each visit of a node in this traversal, NET executes all-pair tree comparisons within its children.

Other work extends the similarity approach by incorporating a variety of additional features such as visual layout information [20] and hyperlinks to detail pages [10]. However, without any assumptions about the target domain, it is difficult to identify such additional features. Moreover, such features are not always available or generally useful. In future work, we plan to extend our method to incorporate additional feature information.

Our method focuses on record extraction and does not extract detailed data in a record. There exist other techniques that address extraction and alignment of attributes in records [19, 21]. Our method can be combined with those techniques to realize the entire data extraction process.

Among existing approaches for template extraction from multiple pages, EXALG [1] is related to our method in its

key idea. EXALG identifies a set of tokens that forms a template based on the intuition that tokens that co-occur with the same frequency within multiple pages are likely to form the same template. Whereas EXALG utilizes occurrence patterns across multiple documents, our method utilizes occurrence patterns within a single document. Thus, the two algorithms are very different.

3. METHODOLOGY

Although automatically identifying and extracting data records from Web pages is considered a hard problem in the computer science community, it is fairly easy for human beings to identify such records. The data records that constitute a Web page are typically represented using an HTML code template. Thus, they often have a similar appearance and are visually aligned. Such a visually repeating pattern can be easily captured by human eyes, and the data records in the visually repeating part can be accurately located. Inspired by this observation, our method comprises three steps: (1) detecting visually repeating information, (2) data record extraction, and (3) semantic-level nesting detection. The first step addresses the problem of what appears repeatedly on the Web page. The second step extracts the data records from the HTML blocks where the repeating patterns occur. The third step extracts the high-level data objects when there is a nested list. The method is fully automatic and does not involve human labeling or feedback. The three steps are described in more detail below.

3.1 Detecting Visually Repeating Information

A *data region* is part of a Web page that contains multiple data records of the same kind, which can be consecutive or non-consecutive. Instead of viewing the Web page as a DOM tree, we consider it as a string of HTML tags. A data region maps to one or more segments of the string with a repeating texture composed of HTML tags, which result in the visually repeating pattern rendered on a Web page. We aim to find the HTML tags that are elements of the data regions.

3.1.1 Visual Signal Extraction

The visual information rendered on a Web page, such as fonts and layout, is conveyed by HTML tags. A given hyperlink tag can have different appearances when it follows different paths in the DOM tree. For each tag occurrence, there is an *HTML tag path*, containing an ordered sequence of ancestor nodes in the DOM tree. Figure 1 shows the different appearances of hyperlink tags defined by two different HTML tag paths.

A Web page can be viewed as a string of HTML tags, where only the opening position of each HTML tag is considered. Each HTML tag maps to an HTML tag path. An example is shown in Table 1. Roughly speaking, each tag path defines a unique visual pattern. Our goal is to mine the visually repeating information in the Web page using this simplified representation.

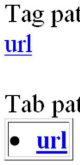


Figure 1: Hyperlinks following different tag paths.

Table 1: Finding tag paths for HTML tags

HTML code	Pos	Tag path
<html>	1	html
<body>	2	html/body
A Web page	3	html/body/h1
<table>	4	html/body/table
<tr>	5	html/body/table/tr
<td> Cell #1 </td>	6	html/body/table/tr/td
</tr>	NA	NA
<tr>	7	html/body/table/tr
<td> Cell #2 </td>	8	html/body/table/tr/td
</tr></table></body> </html>	NA	NA

Table 2: Extracting visual signals from a Web page

Unique tag path	Pos	Visual signal vector
html	1	[1, 0, 0, 0, 0, 0, 0, 0]
html/body	2	[0, 1, 0, 0, 0, 0, 0, 0]
html/body/h1	3	[0, 0, 1, 0, 0, 0, 0, 0]
html/body/table	4	[0, 0, 0, 1, 0, 0, 0, 0]
html/body/table/tr	5,7	[0, 0, 0, 0, 1, 0, 1, 0]
html/body/table/tr/td	6,8	[0, 0, 0, 0, 0, 1, 0, 1]

An inverted index characterizing the mappings from HTML tag paths to their locations in the HTML document can be built for each Web page, as shown in Table 2. Each indexed term in the inverted index, *i.e.*, one of the unique tag paths, is defined to be a visual signal.

Formally, a *visual signal* s_i is a triple $< p_i, S_i, O_i >$, where p_i is a tag path, S_i is a *visual signal vector* that represents occurrence positions of p_i in the document, and O_i represents individual occurrences (*i.e.*, DOM tree nodes). S_i is a binary vector where $S_i(j) = 1$ if p_i occurs in the HTML document at position j and $S_i(j) = 0$ otherwise. O_i is an ordered list of occurrences $(o_i^1, \cdots, o_i^m)$, where o_i^k corresponds to the kth occurrence of 1 in S_i.

Examples of visual signal vectors are shown in the third column of Table 2. All of the visual signal vectors extracted from a Web page have the same length, which is the total number of HTML tag occurrences in the Web page.

The vector representation $\{S_i\}$ of a Web page is much simpler than the DOM tree representation. It also captures how a Web page is organized. Figure 3(a) shows a snapshot of a DBLP [8] Web page containing lists of publication records and other data objects. The extracted visual signals and the visual signal vectors are shown in Figures 3(b) and 3(d). Each row in Figure 3(d) is a visual signal vector. Due to space limitations, we show only the first part of each visual signal vector. The visual signal vectors represent how each atomic-level visual pattern repeats in the Web page. The visually repeating patterns in a Web page involve multiple visual signals. These visual signals together form a certain repeating texture as shown in Figure 3(d). Each texture corresponds to a data region that contains multiple data records of the same kind.

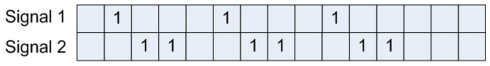
Detecting the visually repeating information is equivalent to identifying the set of visual signals with similar patterns

that are elements of the same data region. In other words, detecting visually repeating information is a *clustering problem*. The visual signals in the same data region are grouped together, while the visual signals not in the same data region are split into different clusters. We use spectral clustering [13] to cluster the visual signals, because of its superior experimental performance and theoretical soundness.

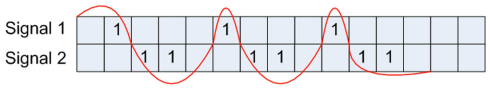
3.1.2 Similarity Measurement

The spectral clustering algorithm produces clustering results based on the pairwise similarity matrix calculated from the data samples. A similarity function captures the likelihood that two data samples belong to the same cluster. A critical factor in determining clustering performance is the choice of similarity function.

In our case, the similarity function captures how likely two visual signals belong to the same data region. Figure 2(a) shows a pair of visual signals that are highly likely to belong to the same data region. Their positions are close to each other, and they interleave with each other. Every occurrence of visual signal 1 is followed by two occurrences of visual signal 2.



(a) A pair of similar visual signal vectors.



(b) Segmented visual signal vectors.

Figure 2: Example pair of visual signals that appear regularly.

The distance between the centers of gravity of two visual signals characterizes how close they appear. We call this measure the *offset* ω and calculate it in Equation (1).

$$\omega(S_i, S_j) = \left| \frac{\sum_{S_i(k)=1} k}{\sum S_i(k)} - \frac{\sum_{S_j(k)=1} k}{\sum S_j(k)} \right| \quad (1)$$

In Equation (1), S_i and S_j are two visual signal vectors and $k \in \{1, 2, ..., l\}$, where l is the length of the visual signal vectors, and $S_i(k)$ is the kth element of S_i.

To capture the interleaving characteristic, we estimate how *evenly* one signal is *divided by* the other. We define *a segment of S_i divided by S_j* as follows: a segment is a (non-empty) set of occurrences of visual signal s_i between any pair of which there is no occurrence of visual signal s_j. Figure 2(b) illustrates how two signals divide each ohter. Let D_{S_i/S_j} be the occurrence counts in the segments of S_i divided by S_j. In our example, $D_{S_1/S_2} = \{1, 1, 1\}$ and $D_{S_2/S_1} = \{2, 2, 2\}$. We define the *interleaving measure* ι in terms of the variances of counts in D_{S_i/S_j} and D_{S_j/S_i} in Equation (2).

$$\iota(S_i, S_j) = \max\{Var(D_{S_i/S_j}), Var(D_{S_j/S_i})\} \quad (2)$$

Both the offset measure and the interleaving measure yield non-negative real numbers. A smaller value of either measure indicates a high probability that the two visual signals come from the same data region. The *similarity measure* $\sigma(s_i, s_j)$ between two visual signals is inversely proportional to the product of these two measures and is defined by Equation (3).

$$\sigma(s_i, s_j) = \frac{\varepsilon}{\omega(S_i, S_j) \times \iota(S_i, S_j) + \varepsilon} \quad (3)$$

In Equation (3), ε is a non-negative term that avoids dividing by 0 and that normalizes the similarity value so that it falls into the range $(0, 1]$. In our experiments, we chose $\varepsilon = 10$.

Given Equation (3), we can calculate the similarity value of any pair of visual signals. Example results are shown in Figure 3(c). The pixel in the ith row and jth column shows the similarity value for visual signal s_i and visual signal s_j. A bright pixel indicates a high similarity value, whereas a dark pixel indicates a low similarity value. Thus, the visual signals in Figure 3(b) aligned with the large bright blocks in Figure 3(c) are likely to be from the same data region. The visual signals actually involved in the data regions (*i.e.*, the *ground truth*) are highlighted in rectangles. Each box includes one data region. As expected, the similarity measure captures the likelihood of two visual signals being from one data region.

3.1.3 Visual Signal Clustering

The pairwise similarity matrix can be fed into a spectral clustering algorithm directly. We employ the normalized cut spectral clustering algorithm developed by Shi *et al.* [14] to produce the groups of visual signals with similar patterns. A cluster containing n visual signals indicates that those n visual signals are from the same data region with high probability.

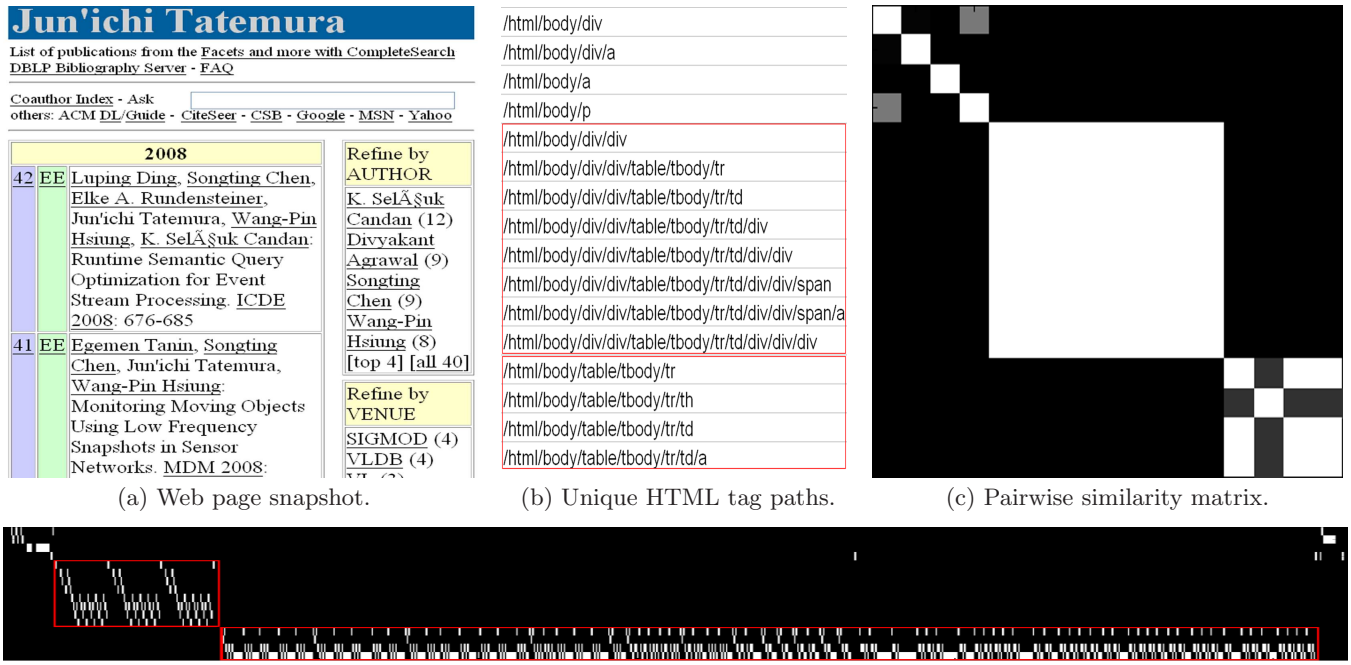
A data region contains multiple data records that use the same HTML code template, and a template typically has multiple HTML tags that differentiate the data attributes. The *size of a template* is defined to be the number of unique HTML tag paths involved in the template. Thus, a template with size greater than n should correspond to a cluster containing more than n visual signals. Given the fact that most HTML code templates contain more than three HTML tags that differentiate different data attributes, we assume that the smallest size of a template is three. Thus, we need to examine only the clusters containing three or more visual signals. We call these clusters the *essential clusters* of the document, and denote them by $\mathcal{C} = \{C_1, \cdots, C_m\}$.

In the example shown in Figure 3, there are two clusters of size greater than three produced by the spectral clustering algorithm. These clusters correspond to the *ground truth*, *i.e.*, match the visual signals involved in the data regions exactly, as shown in Figure 3(b), where each cluster corresponds to one data region and contains a set of homogenous data records, as shown in Figure 3(a).

3.2 Data Record Extraction

Visual signals that are grouped together in an essential cluster $C \in \mathcal{C}$ should represent the same data region. Each occurrence o_i^k of visual signal s_i in C represents part of a data record, an entire data record, or a set of data records. The goal of data record extraction is to identify occurrences that represent individual data records.

To find such occurrences, we introduce ancestor and descendant relationships between visual signals. We say that s_i is an *ancestor* of s_j, denoted by $s_i // s_j$, iff p_i is a prefix of p_j. For example, */html/body/p* is an ancestor of



(a) Web page snapshot. (b) Unique HTML tag paths. (c) Pairwise similarity matrix.

(d) Visual signal vectors. Each row is a visual signal vector. Bright pixels correspond to 1s and dark pixels correspond to 0s.

Figure 3: Pairwise similarity matrix calculated based on Equation (3).

$/html/body/p/a$. We also employ the standard relationships between occurrences o_i and o_j by viewing them as DOM nodes: $o_i//o_j$ (o_i is an ancestor of o_j in a DOM tree), and $o_i < o_j$ (o_i is a predecessor of o_j in the document order).

If $s_i//s_j$ then, for each $o_j \in O_j$, there exists $o_i \in O_i$ such that $o_i//o_j$, meaning that the HTML region represented by s_i contains the region represented by s_j. Recall that, if s_i and s_j are clustered together, they are in the same data region. Thus, an ancestor visual signal s_i is more likely to represent a set of entire data records while a descendant visual signal s_j is more likely to represent a set of data attributes.

Among the visual signals in an essential cluster C, there is at least one visual signal that has no ancestor in C. We call these visual signals the *maximal ancestor visual signals*. The occurrences of maximal ancestor visual signals are considered first in data record extraction because they are more likely to be individual data records. We discuss below how to find the exact data record boundaries in two different scenarios.

3.2.1 Single Maximal Ancestor Visual Signal

If there is only one maximal ancestor (say s_m) in an essential cluster C, the occurrences o_m^i are likely to be individual data records. However, note that:

1. *Not all of the occurrences are data records.* Recall that o_m^i is one of the occurrences of tag path p_m (*e.g.*, $/html/body/p$). This path may be used for representing not only data records but also different regions. Thus, we need to exclude occurrences that are used for different purposes based on the following intuition: A data record should consist of not only an occurrence of s_m but also occurrences of other visual signals in C (that are descendants of s_m).
2. *An occurrence can contain multiple data records.* For example, product information on an e-commerce Web page might be organized in multiple columns (*e.g.*, Figure 5(a)). Let s_R and s_P be visual signals that represent rows of the product list and individual product records, respectively. They are grouped together into C. Because $s_R//s_P$, s_R is the maximal ancestor and, thus, we identify occurrences of s_P as data records.

To address the above issues, we introduce the techniques of record candidate filtering and record separation.

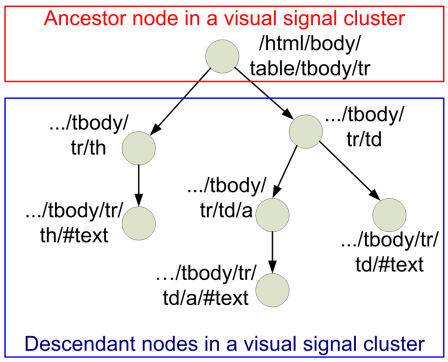
Record candidate filtering. Record candidate filtering selects occurrences from O_m that contain data records. The intuition is as follows: If o_m^i has many descendants that are occurrences of other visual signals in C, o_m^i is likely to contain data records. Let $D_m^i (\subset C)$ be a set of visual signals that have occurrences in the descendants of o_m^i. A greater value of $|D_m^i|$ indicates that o_m^i is a record candidate. We assume further that not all of the visual signals in C are equally important. If a visual signal appears in every data record, it has high similarity to other visual signals in C. Thus, we introduce a weighting factor for each visual signal s_j in D_m^i based on its intra-cluster similarity in C and define the record candidate score ρ of an occurrence o_m^i by:

$$\rho(o_m^i) = \sum_{s_j \in D_m^i} \sum_{s_k \in C} \sigma(s_j, s_k) \quad (4)$$

where $\sigma(s_j, s_k)$ is be calculated using Equation (3).

We filter out o_m^i iff $\rho(o_m^i) < \rho_{max} \times \alpha$, where ρ_{max} is the maximum ρ score of occurrences of all the visual signals in C. In our experiments, we chose $\alpha = 0.5$.

To identify D_m^i, we need to check if there is an occurrence o_j of a visual signal s_j such that $o_m^i//o_j$ for each s_j in C. Note that $s_m//s_j$ because s_m is the only maximal ancestor



(a) Recovered ancestor and descendant relationships within one cluster.

(b) Data record extraction results after filtering out the occurrences of the common ancestor visual signal.

Figure 4: Maximal ancestor visual signal containing one data record.

in C. Thus, we only need to check if $o_m^i < o_j < o_m^{(i+1)}$ to write $o_m^i // o_j$, which is done efficiently using the visual signal vectors S_m and S_j.

Record separation. If the occurrences of the maximal ancestors contain multiple data records, their direct descendant should be able to better separate the data records. We examine the DOM subtrees of the occurrences to determine whether the child nodes are more likely to be individual data records. First, they must be occurrences of the same visual signal. Next, they must have a similar visual pattern so that together they comprise a large visually repeating block. This idea is similar to one employed in MDR [11] that checks if a single row contains multiple data records. Whereas MDR utilizes edit distance of tag structures, our method takes a simpler approach that performs well in experiments. From the rendered Web page, we retrieve the width and height of all of the descendant visual signal occurrences. We calculate their variances to determine whether the descendant node is a better data record separator.

The record filtering and separation are performed repeatedly until no better separator is found. The results are the atomic-level data records in a Web page.

Figure 4 shows an example where the maximal ancestor represents the data record and no record separation is required. In the DBLP page, the publications of a researcher are listed in a table and all of them are extracted correctly. An example that requires record separation is shown in Figure 5. In this example, each row contains two product records. Our algorithm extracts the visual signal corresponding to a row as the maximal ancestor and then determines whether its direct descendant visual signal is a better record separator.

3.2.2 *Multiple Maximal Ancestor Visual Signals*

When there are multiple maximal ancestors, there is no single visual signal that captures the entire data record. Typically, occurrences of these different maximal ancestors are consecutive siblings that together represent a data record.

Our problem now is to identify a repeating pattern from a sequence of occurrences from different signals. Our current implementation uses a simple heuristic: The visual signal, say s_B, that occurs first is chosen as the record boundary. The intuition is that the first component of a record is typically a mandatory part of the data (*e.g.*, a title). An occurrence o of other maximal ancestor visual signals is a part of the ith data record if $o_B^i < o < o_B^{(i+1)}$. After forming the data record candidates, we filter them as in Section 3.2.1.

3.3 Semantic-Level Nesting Detection

Nested lists are common on the Web. Usually, data records are organized into semantic categories with an arbitrary number of data records in each category. A description might be attached to each category.

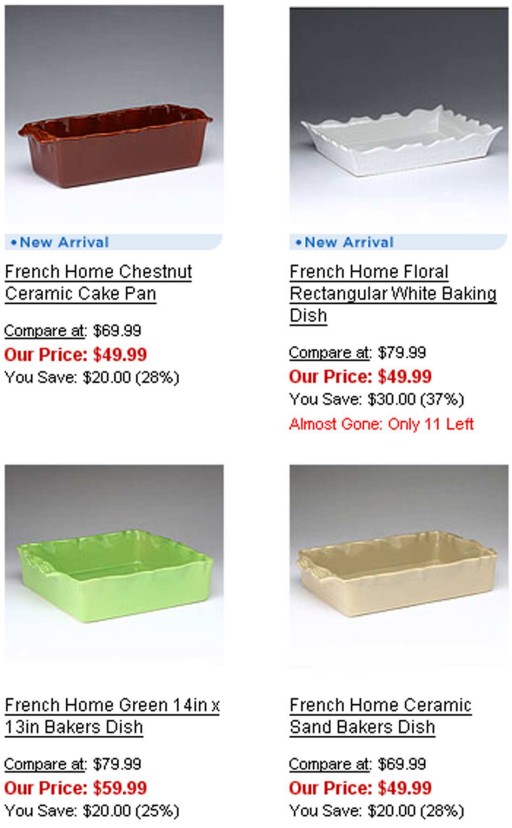
Our approach can capture such nesting through discovery of non-consecutive lists of atomic-level data records. The semantic categories are usually explicitly marked by HTML tags, and data records inside one semantic category are consecutive in the HTML document. Thus, if the data records are not consecutive, they might belong to different semantic categories. Based on this intuition, we extract the nesting structure as follows: If a visual signal occurs at each point where the same set of data records is partitioned, the visual signal corresponds to a visual pattern that separates two semantic categories. The text lying between the sets of extracted data records is the description of the semantic category. Using this rule, we extract both the "year" objects and the "publication" objects in the DBLP page example, as shown in Figure 6.

4. EXPERIMENTS

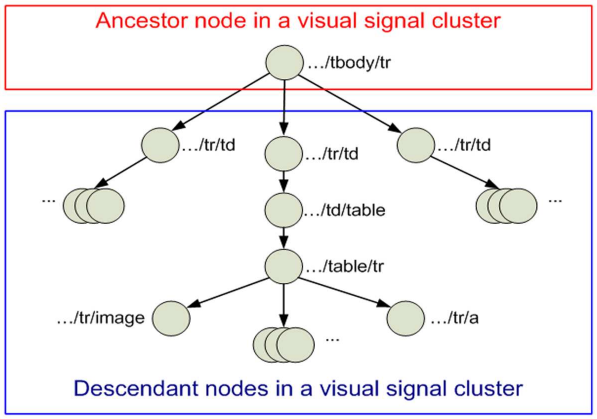
4.1 Experimental Setup

We evaluated both the effectiveness and the efficiency of our algorithm using two data sets. We compare the performance of our algorithm with that of MDR, an implementation of which was available on the Web. Implementation of NET was not available, and EXALG is applicable only when multiple Web page instances with the same structure are available. Thus we do not compare the performance of our algorithm with that of those algorithms.

Data set #1 was chosen from the testbed for information extraction from the deep Web, collected by Yamada *et al.*



(a) Web page snapshot.



(b) Recovered ancestor/descendant relationships within one cluster.

(c) Data record extraction results.

Figure 5: Maximal ancestor visual signal containing multiple data records.

Nested objects set #2

Description	Repeating Item Sets
2008	[4]
2007	[5]
2006	[6]
2005	[7]
2003	[8]
2000	[9]
1999	[10]
1997	[11]
1994	[12]
1993	[13]
1992	[14]
1991	[15]
1989	[16]

(a) Nested objects.

Repeating item set #4 2008

Record #1 42 EE Luping Ding, Songting Chen, Elke A. Rundensteiner, Jun'ichi Tatemura, Wang-Pin Hsiung, K. SelÃ§uk Candan: Runtime Semantic Query Optimization for Event Stream Processing. ICDE 2008: 676-685

Record #2 41 EE Egemen Tanin, Songting Chen, Jun'ichi Tatemura, Wang-Pin Hsiung: Monitoring Moving Objects Using Low Frequency Snapshots in Sensor Networks. MDM 2008: 25-32

Record #3 40 EE Jun'ichi Tatemura, Songting Chen, Fenglin Liao, Oliver Po, K. SelÃ§uk Candan, Divyakant Agrawal: UQBE: uncertain query by example for web service mashup. SIGMOD Conference 2008: 1275-1280

Record #4 39 EE Yan Qi, K. SelÃ§uk Candan, Jun'ichi Tatemura, Songting Chen, Fenglin Liao: Supporting OLAP operations over imperfectly integrated taxonomies. SIGMOD Conference 2008: 875-888

Record #5 38 EE Yu-Ru Lin, Hari Sundaram, Yun Chi, Jun'ichi Tatemura, Belle L. Tseng: Detecting splogs via temporal dynamics using self-similarity analysis. TWEB 2(1): (2008)

Repeating item set #5 2007

Record #1 37 EE Yu-Ru Lin, Hari Sundaram, Yun Chi, Jun'ichi Tatemura, Belle L. Tseng: Splog Detection Using Self-similarity Analysis on Blog Temporal Dynamics. AIRWeb 2007

Record #2 36 EE Yun Chi, Shenghuo Zhu, Xiaodan Song, Jun'ichi Tatemura, Belle L. Tseng: Structural and temporal analysis of the blogosphere through community factorization. KDD 2007: 163-172

Record #3 35 EE Jun'ichi Tatemura, Arsany Sawires, Oliver Po, Songting Chen, K. SelÃ§uk Candan, Divyakant Agrawal, Maria Goveas: Mashup Feeds: : continuous queries over web services. SIGMOD Conference 2007: 1128-1130

(b) Atomic-level objects.

Figure 6: Data record extraction result for nested lists.

[18]. The testbed data has 253 Web pages from 51 Web sites randomly drawn from 114,540 Web pages with search forms. The data records in these Web pages are manually labeled; the results are available online together with the testbed data set. To provide a fair comparison between our algorithm and the MDR algorithm [11], which is designed for flat data records, we filtered out the Web pages with nested structures in the testbed. The resulting data set #1 contains 213 Web pages from 43 Web sites.

Data set #2 was introduced mainly for the purpose of evaluating our algorithm on nested list structures. Lacking an existing test data set, we collected the Web pages ourselves. Data set #2 contains 45 Web pages, each from one Web site, randomly chosen from the domains of business, education, and government. Each Web page contains a two-level nested list structure. Both the atomic-level data records and the nested data records are manually labeled.

Our experiments were carried out on a Pentium 4 computer with a 3.2GHz CPU and 2G of RAM. Our Java implementation of the algorithm utilizes the open source Web renderer Cobra [6], which resolves ill-formed HTML and executes JavaScript for dynamic HTML pages.

4.2 Accuracy Analysis

The experimental results for our algorithm compared with MDR [11] are shown in Figure 7. We ran both algorithms for all of the Web pages in data set #1. The results are aggregated based on the Web sites. The *ground truth* is the set of data records in all Web pages from one Web site. *True positives* are the set of data records correctly extracted by the algorithms from that Web site. The perfect case is that the true positives match the ground truth exactly. *False positives* are the set of data records that the algorithm incorrectly includes in the same list with the true positives. To distinguish the false positives from the true positives, we flip the sign of the false positives and show them in the same figure. Generally speaking, our algorithm has more true positives and fewer false positives compared with the MDR algorithm. We also calculated the *precision* and *recall* as given in Equations (5) and (6) for all of the Web sites. The results are shown in Table 3. When none of the records is detected, both $|true\ positives|$ and $|false\ positives|$ are zero; hence, Equation (5) is ill-formed. We omit such cases to calculate the average precision. The numbers in square brackets are the average precision of both algorithms when we define the precision to be zero in such a case.

$$Precision = \frac{|true\ positives|}{|true\ positives| + |false\ positives|} \quad (5)$$

$$Recall = \frac{|true\ positives|}{|ground\ truth|} \quad (6)$$

Table 3: Accuracy comparison for data set #1

Algorithm	Average Precision	Average Recall
Our algorithm	96.2% [90.4%]	93.1%
MDR	93.2% [59.8%]	61.8%

The experimental results for data set #2 show the performance of our algorithm for nested list structures. We compare each atomic-level data record and nested data record extracted by our algorithm with the manually labeled ground truth. The results of the comparison are shown in Table 4. The ground truth numbers of data records for the Web pages are listed in columns 2 and 3. The numbers of true positives are listed in columns 4 and 5. The numbers of false positives are listed in parentheses if they are greater than zero. There are 15 Web pages from the business domain, 15 Web pages from the education domain, and 15 Web pages from the government domain.

Table 4: Experimental results for data set #2

Domain	Ground Truth		Our Results	
	Nested	Atomic	Nested	Atomic
Business	46	415	46	415(1)
Education	215	1672	208(2)	1672(17)
Government	104	955	104(1)	954(1)

Overall Accuracy Measure				
Nested Records	Precision	98.9%	Recall	98.1%
Atomic Records	Precision	99.4%	Recall	99.9%

4.3 Time Complexity Analysis

The algorithm consists of three steps. We analyze the time complexity for each step individually.

Detecting visually repeating information. In this step, first we scan the Web page and extract the visual signals, which takes $\mathcal{O}(L)$ time, where L is the total number of HTML tag occurrences in the Web page. Calculating the pairwise visual signal similarity matrix and performing spectral clustering on it takes $\mathcal{O}(M \times L) + \mathcal{O}(M^3)$, where M is the number of unique HTML tag paths in the Web page. Thus, the step of visual repeating information detection takes $\mathcal{O}(M \times L) + \mathcal{O}(M^3)$ time in total.

Data record extraction. In this step, first we retrieve all of the occurrences of the common ancestors in the Web page for each essential cluster. When filtering these occurrences, the algorithm visits all of the descendants. The total number of HTML nodes visited is less than L. Thus, the time complexity of this step is $\mathcal{O}(L)$.

Semantic-level nesting detection. In this step, we examine the visual signals that appear at each point where the data records are not consecutive. The number of HTML tags visited is still less than L. Thus, the time complexity of this step is $\mathcal{O}(L)$.

In total, the time complexity of the algorithm is $\mathcal{O}(M \times L) + \mathcal{O}(M^3)$, where L is the total number of tag occurrences and M is the number of unique tag paths in the Web page.

For comparison purposes, we also analyze the time complexity of existing similarity-based approaches, MDR[11] and NET[12]. These algorithms traverse a DOM tree and apply edit distance computation between sibling subtrees. Let N be the number of children of each node. At the root, the algorithms compute the edit distance between its children with size L/N, taking $\mathcal{O}((L/N)^2)$ time. MDR computes the edit distance N times, and NET computes it N^2 times in the worst case. At depth d, there are N^d trees, each of which has N children of size L/N^{d+1}. The total cost is $\sum_d (L/N^{d+1})^2 N^k N^d = L^2 N^{k-2} \sum_d (1/N)^d < L^2 N^{k-2} \times N/(N-1)$ where $k = 1$ for MDR and $k = 2$ for NET. Thus, the time complexity of MDR and NET are $\mathcal{O}(L^2/N)$ and $\mathcal{O}(L^2)$, respectively. From this analysis, we conclude that MDR is efficient ($\mathcal{O}(L)$) when the document

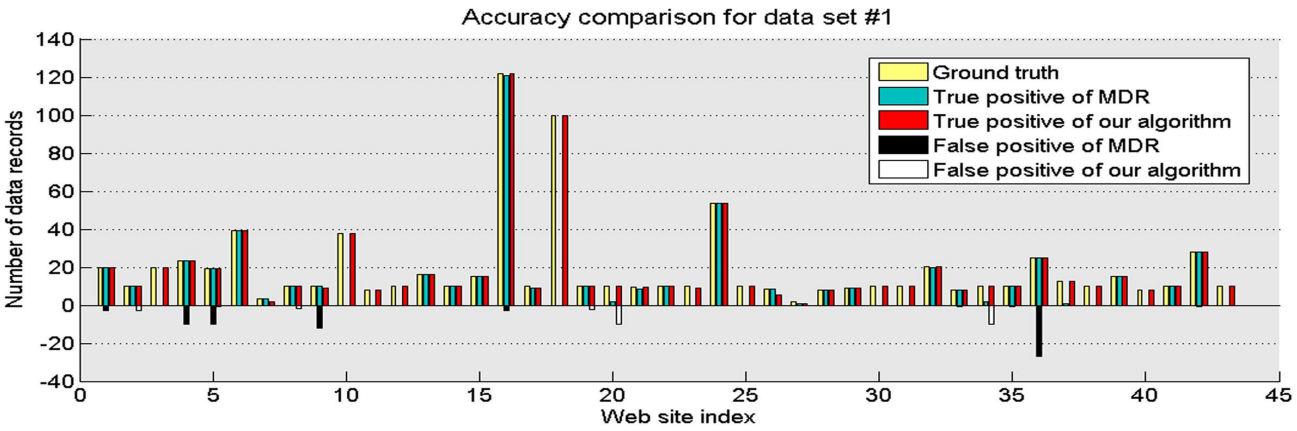


Figure 7: Accuracy comparison between our algorithm and MDR for data set #1.

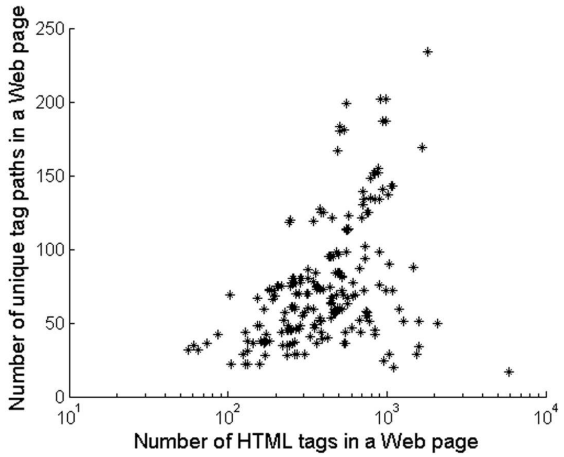


Figure 8: The number of unique tag paths does not increase as the number of HTML tags increases.

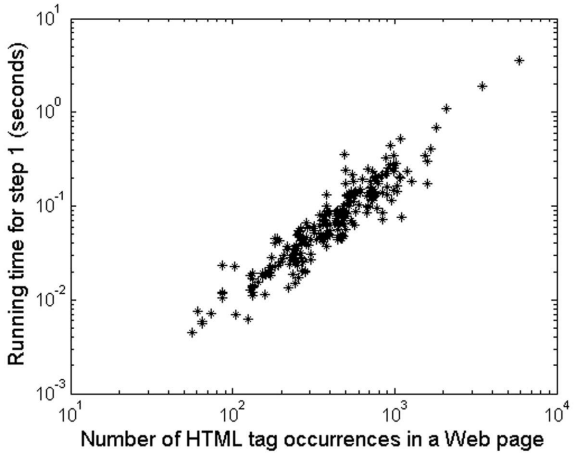


Figure 9: Step 1 is linear in the document length.

structure is simple (and N is as large as L). However, if the document structure is complex, MDR is not as scalable.

The key question then is how the number M of unique tag paths, grows as L becomes large. If M does not scale up, our algorithm is more scalable than NET, and even MDR when the document is complex. Recall that our algorithm and NET can detect nested structures whereas MDR cannot. For the experimental data set, M stays small as L grows, as shown in Figure 8. Thus, the complexity of our algorithm is $\mathcal{O}(L)$ for practical data sets, *i.e.*, it is linear in the document length. Figure 9 shows that the completion time of Step 1 is linear in L.

On average, the total execution time of our algorithm for one Web page is similar to the rendering time. We divide the execution time into three parts based on the three steps as presented in Table 5. Step 1 takes 47.95% of the total time, and Steps 2 and 3 together take 52.05% of the total time. Because Steps 2 and 3 are conducted for each essential cluster, and there is no interaction between clusters, this part of the algorithm can be parallelized.

Table 5: Execution time analysis

Function	Average Time (ms)	Percentage
Rendering	208.90	NA
Total execution time	328.73	100%
Step 1	157.63	47.95%
Step 2	72.99	22.20%
Step 3	98.11	29.84%

5. CONCLUSION AND FUTURE WORK

This paper presents a novel approach to data record extraction from Web pages. The method first detects the visually repeating information on a Web page and then extracts the data records. The notion of visual signal is introduced to simplify the Web page representation as a set of binary visual signal vectors instead of the traditional DOM tree. A data record list corresponds to a set of visual signals that appear regularly on the Web page. The normalized cut spectral clustering algorithm is employed to find the visual signal clusters. For each visual signal cluster, data record extraction and nested structure detection are conducted to extract both atomic-level and nested-level data records.

Experimental results on flat data record lists are compared with a state-of-the-art algorithm. Our algorithm shows significantly higher accuracy than the existing work. For data record lists with a nested structure, we collected Web pages from the domains of business, education, and government. Our algorithm demonstrates high accuracy in extracting both atomic-level and nested-level data records. The execution time of the algorithm is linear in the document length for practical data sets.

The algorithm depends only on the Web page structure without examining the Web page content, which makes it a domain-independent approach. The algorithm is suitable for handling Web-scale data because it is fully automatic and does not need any information other than the Web page.

In the future, we will extend this work to support data attribute alignment. Each data record typically contains multiple data attributes. Unfortunately, there is no one-to-one mapping from the HTML code structure to the data record structure. Identification of the data attributes offers the potential of better use of the Web data.

The work presented here extracts data records from single Web pages. However, the World Wide Web is composed of billions of Web pages each with their own data records. Future work will include integration of heterogeneous data records across different Web pages.

6. ACKNOWLEDGEMENT

The authors wish to thank Professor P. M. Melliar-Smith and Professor D. Agrawal for their valuable help during this research.

7. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from Web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on the Management of Data*, pages 337-348, 2003.

[2] D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the World Wide Web. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems*, pages 361-370, 2001

[3] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the power of tables on the Web. In *Proceedings of the 34th International Conference on Very Large Data Bases*, pages 538-549, 2008.

[4] C. Chang and S. Lui. IEPAD: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 681-688, 2001

[5] K. C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the Web: Observations and implications. *ACM SIGMOD Record*, 33(3):61-70, 2004.

[6] Cobra: Java HTML Renderer and Parser, `http://lobobrowser.org/cobra.jsp`.

[7] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109-118, 2001.

[8] DBLP Computer Science Bibliography, `http://www.informatik.uni-trier.de/~ley/db/`.

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

[10] K. Lerman, L. Getoor, S. Minton, and C. Knoblock. Using the structure of Web sites for automatic segmentation of tables. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 119-130, 2004.

[11] B. Liu. Mining data records in Web pages. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 601-606, 2003.

[12] B. Liu and Y. Zhai. NET: System for extracting Web data from flat and nested data records. In *Proceedings of the Conference on Web Information Systems Engineering*, pages 487-495, 2005.

[13] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the Neural Information Processing Systems Conference*, pages 849-856, 2001.

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888-905, 2000.

[15] K. Simon and G. Lausen. ViPER: Augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 381-388, 2005.

[16] J. Tatemura, S. Chen, F. Liao, O. Po, K. S. Candan, and D. Agrawal. UQBE: Uncertain query by example for Web service mashup. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1275-1280, 2008.

[17] J. Wang and F. H. Lochovsky. Data extraction and label assignment for Web databases. In *Proceedings of the 12th International Conference on the World Wide Web*, pages 187-196, 2003.

[18] Y. Yamada, N. Craswell, T. Nakatoh, and S. Hirokawa. Testbed for information extraction from deep Web. In *Proceedings of the 13th International Conference on the World Wide Web*, pages 346-347, 2004.

[19] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on the World Wide Web*, pages 76-85, 2005.

[20] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th International Conference on the World Wide Web*, pages 66-75, 2005.

[21] H. Zhao, W. Meng, and C. Yu. Mining templates from search result records of search engines. In *Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining*, pages 884-893, 2007.

[22] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. Simultaneous record detection and attribute labeling in Web data extraction. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining*, pages 494-503, 2006.