Instance-based Schema Matching for Web Databases by Domain-specific Query Probing

Abstract

In a Web database that dynamically provides information in response to user queries, there are two distinguishing schemas, interface schema and result schema, presented to users. Each of them partially reflect schema of the backend database. Most previous works merely studied the problem of schema matching across query interfaces of Web databases. In this paper, we propose a novel schema model that, in particular, distinguishes the interface schema (the schema users can query) and the result schema (the schema users can browse) of a Web database in a specific domain. In this model, we address two significant schema matching problems for Web databases, intra-site schema matching and inter-site schema matching. The first problem is crucial in automatically extracting data from Web databases, while the second problem plays a significant role in meta-retrieving and integrating data from different Web databases. We also investigate the feasibility of a unified solution to the two problems based on query probing and instance-based schema matching techniques. Benefiting form the model, a cross validation technique is also proposed to improve the accuracy of various schema matchings. Our experiments on real Web databases demonstrate that the two problems can be solved at the same time with high precision and recall.

1. Introduction

The Web is a huge information repository and is growing at a prodigious rate. Besides web pages accessible or crawlable by specific URLs, the Web also contains a vast amount of non-crawlable content. This *hidden* part of the Web, referred to as the *deep Web* [5] or the *hidden Web* [14], is comprised of a large number of online *Web databases*. An online Web database consists of a searchable interface (usually an HTML form) and a backend database, which dynamically provides information in response to user queries. As compared to the static surface Web, Web databases contain a much larger amount of high-quality (often structured) information [8].

In the deep Web, it is usually difficult or even impossible to directly obtain the schemas of the websites' backend databases without cooperation from the sites. Instead, the sites present two other distinguishing schemas, *interface schema* and *result schema*, to users (e.g., the website in Figure 1). The interface schema is the schema of the query interface of a deep website, which exposes attributes that can be queried in the backend database. The result schema is the schema of the query results, which exposes attributes that are shown to users. The interface schema is useful for applications, such as a mediator that queries multiple Web databases, since the mediator needs complete knowledge about the search interface of each database. The result schema is critical for applications, such as data extraction, where instances in the query results are extracted. In addition to the importance of the interface schema and result schema themselves, attribute matching[1] across different schemas is also important. First, matching between different interface schemas and matching between different results schemas (*inter-site schema matching*) is critical for meta-searching and data-integration among related Web databases. Second, matching between the interface schema and the result schema of a single Web database (*intra-site schema matching*) enables automatic data annotation and database content crawling. Therefore, in this paper we focus on automatically discovering both the interface schemas and the result schemas of Web databases and matching semantically-related attributes between them.

Previous approaches ([16], [17], [21]) to matching the schemas of Web databases primarily focus on matching query interfaces (i.e., on inter-site interface schema

Proceedings of the 30th VLDB Conference,
Toronto, Canada, 2004

[1] Attribute matching is the process of determining the semantic correspondences among the attributes of two schemas.

matching). The basic idea is to identify attribute labels from the descriptive text surrounding interface elements and then find *synonym relationships* between the identified labels. The performance of these approaches may be affected when no attribute description can be identified or the identified description is not informative (e.g., "Search" in the homepage of Amozon.com). In contrast, in this paper we propose, a novel instance-based schema matching approach motivated by the necessity to identify the result schemas of Web databases that often lack available attribute names or labels, and the goal of simultaneously solving inter-site and intra-site schema matching.

Our approach is mainly based on three observations in Web databases. First, improper[2] queries often cause search failure or no returned results. Second, the keywords of proper queries that return a result web page, very likely reappear in the returned results' corresponding attributes. Third, there is an underlying *global schema*[3] for related Web databases in the same domain (proposed and verified in [16]). Accordingly, we introduce a query probing technique that first exhaustively sends query keywords residing in a domain-specific global schema, whose semantics are known in advance, then analyzes the re-occurrences of submitted query terms in the returned result data, and finally identifies the semantically corresponding attributes from both interface schema and result schema from the previous analysis.

By introducing a domain-specific global schema, a combined schema model is presented in this paper to describe five kinds of schema matching for Web databases in the same domain: global-interface matching, global-result matching, interface-result matching, interface-interface matching, and result-result matching. This model not only describes the matching relationships among different schemas of Web databases in a specific domain, but, more importantly, also provides a global view about how to reinforce the matching accuracy by conducting multiple kinds of schema matching simultaneously. In this paper, we also present a cross validation technique to improve the accuracy of the schema matching results.

The main contributions of this paper can be summarized as follows:

- Introduction of a novel schema model of a single Web database, which as far as we know is the first model to distinguish what information users can query and what information users can browse in a Web database.
- Introduction of a generative view that includes five kinds of schema matching for related Web databases in a specific domain.
- Introduction of an instance-based method based on domain-specific query probing, along with mutual information and vector similarity analysis, to automatically match various schemas of Web databases (intra-site and inter-site).
- Introduction, benefiting from the above generative view, of a cross validation technique based on an approximate solution of graph partitioning problem to improve the accuracy of different kinds of schema matching.

The rest of this paper is organized as follows. In section 2, we present the model with five schema matching for Web databases. In section 3, the domain-specific query probing technique is introduced. We propose, in section 4, an instance-based schema matching approach with a cross validation technique, to solve both the intra-site and inter-site schema matching problems at the same time. Section 5 presents the experimental results of testing our approaches on real Web databases. Section 6 reviews existing work on the schema matching problem and how it correlates with our framework. Finally, we give our conclusion and future work in section 7.

2. Combined Schema Model

A Web database is often comprised of a query interface and a backend database. When a user query is submitted into the query interface, the site accesses its backend database for relevant data and returns the results to the user. Specifically, the interface of the Web database usually contains multiple input elements, each of which may be associated with an attribute of the schema of the backend database. Data objects that the Web database returns to users are usually *semi-structured*, as their attribute values are encoded into HTML tags. Therefore, both the interface and returned results of the Web database partially reflect the same schema of the backend database in different ways.

For instance, Figure 1 shows an example of an online bookstore[4]. A possible schema of the backend database is shown in the middle, consisting of at least six[5] attributes {Title, Author, Publisher, ISBN, Format, Publication Date}. The query interface, shown on the left, contains five input elements with surrounding text describing their semantics. When a keyword query "Harry Potter" is submitted into the "Title" element in the interface, a result page is returned by the web site containing its answer to the query. In Figure 1, a part of the result page is shown on the right and it contains three book instances with associated attribute values.

[2] "Proper" means that the semantics of query terms match the semantics of the input element.

[3] The global schema is a view capturing common attributes of data in the specific domain.

[4] http://www.mysimon.com/

[5] The exact number is not known.

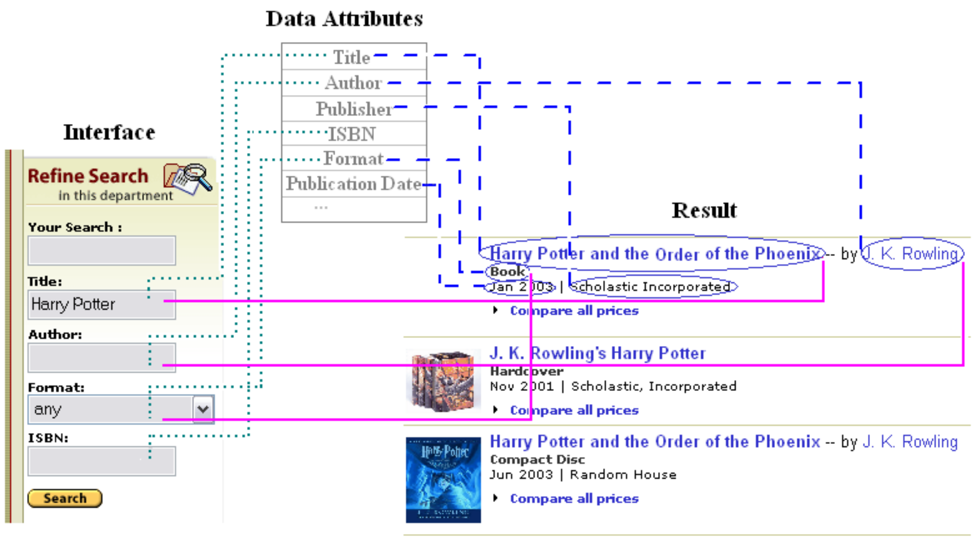


Figure 1. An example of a Web database with its search interface and result page.

From this example, we can clearly see the difference between the attribute information contained in the query interface and that contained in the result pages. Although the site provides an element in the interface for users to search on a particular data attribute, data values of this attribute may not appear in a result page. Likewise, the returned results may have some attributes that users cannot query in the interface. For instance, the site provides search functionality in the interface over the "ISBN" attribute while no ISBN information is included in the result page. At the same time, there is some publisher information for each book instance in the result page, while the site does not provide search functionality over the "Publisher" attribute. Furthermore, three kinds of semantic correspondence are shown in Figure 1 represented by different style lines (dotted, dashed and solid). They are respectively, the correspondence between data attributes of the primary schema and elements in the interface, the correspondence between the data attributes and instance values in the result pages, and the correspondence between elements in the interface and instance values in the result pages.

To describe the various schemas of Web databases and the attribute matching among them, a combined 3-layer schema model is necessary. However, in the deep Web, the primary schema of a backend database is hard to obtain directly as it is hidden behind search interfaces. Instead, previous work [16] makes the significant observation that there exists an underlying generative global schema that can be discovered for related Web databases in a specific domain, by examining the query interfaces of Web databases. Thus, we introduce a global schema (i.e., a view capturing common attributes of data in the specific domain.) to substitute for the primary schema of the backend database and propose a combined model for matching schemas of Web databases. Besides its availability, another advantage of introducing a global schema is that it simplifies the process of matching schemas of different Web databases in the same domain as they share the same global schema.

Formally, we define a *schema* as a set of attributes, each of which corresponds to some unique meaning. In our model, the Web databases can be categorized into a number of domains, where Web databases in the same domain provide information about the same type of product (e.g., Books or Used cars) or the same topic (e.g., Jobs). In each specific domain, there exists a unified *global schema* (**GS**) representing the common knowledge about the domain. In addition, each Web database in this model consists of two different schemas, the *interface schema* (**IS**) and the *result schema* (**RS**) (illustrated in Figure 2 as nodes). In particular, the global schema consists of the representative attributes of the data objects in this domain. The interface schema of an individual Web database consists of data attributes over which users can query, while the result schema consists of data attributes that users can browse in the Web database. The three schemas of a Web database all partially represent the data objects contained in the backend database, varying only on the number of attributes and attribute names.

A *matching* between two schemas S_1 and S_2 determines that certain attributes of schema S_1 semantically correspond to certain attributes of schema S_2. For an individual Web database, there exist three kinds of *intra-site schema matching*, between GS and IS, between GS and RS, and between IS and RS (illustrated in Figure 2 as edges between heterogeneous nodes of each single site). Furthermore, given multiple Web databases in the same domain, the interface schemas of different Web databases can also be pair-wise matched (between IS and IS), as can the result schemas of different Web databases

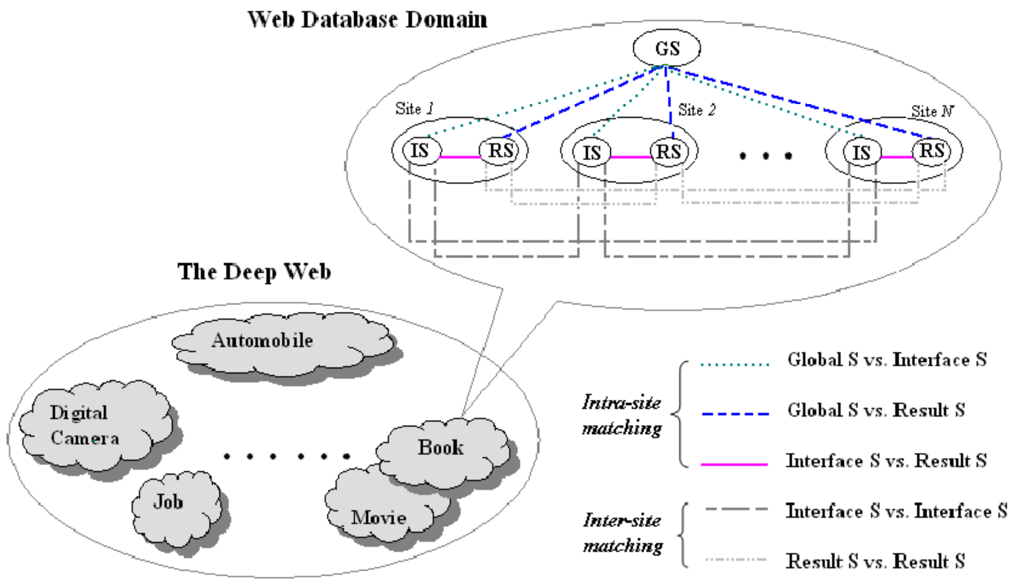


Figure 2. Global view of the Deep Web and combined schema model of Web databases.

(between RS and RS). Such *inter-site schema matching* is illustrated in Figure 2 as dashed edges between homogenous nodes of different sites.

The benefits of such a model are that it allows us to:

- **Automatically understand the semantics of schema attributes.** In this model, if the attribute semantics of one particular schema are accurately identified or pre-known, then the attribute semantics of other schemas in this model can also be discovered as long as they are correctly matched to an identified one. Even if the semantics of one particular schema is somehow wrongly identified in a matching with a particular schema, there is still opportunity for correction when it is matched to other homogeneous schemas in this model.
- **Automatically extract relevant content from Web databases.** Crawling the massive information hidden behind search interfaces of Web databases is a major problem for the Web search community. Automatic understanding of interface schemas can make it possible for crawlers to intelligently submit "appropriate" queries into the right input elements. Furthermore, automatic understanding of result schemas can make it possible for crawlers to intelligently obtain valid query results according to their semantic (i.e., to automatically extract relevant Web database content).
- **Meta-search multiple Web databases.** In this model, related Web databases are categorized by their domains. With a meta-search interface built for each domain, users can search multiple Web databases of the domain at one time. Given a user query, first some promising Web databases that may contain relevant information are picked, and then queries are sent to these Web databases according to the identified semantics of their search interfaces. Finally, their query results are integrated and displayed to users according to the match among their result schemas.

In this paper, the first benefit is our focus and achieved by matching both interface schemas and result schemas of Web databases in the same domain to a domain-specific global schema, in order to discover their attribute semantics. We also introduce a cross validation technique to improve the accuracy of various matching.

3. Domain-specific Query Probing

Database schema matching is the task of finding mappings between attributes of two schemas that semantically correspond to each other [3]. Previous approaches to schema matching can be categorized into two classes; *label-based* and *instance-based*, according to the different information they rely on (see [22] for a survey). Label-based methods only consider the similarity between schema definitions or attribute labels of two databases. Instance-based methods ([13] and [19]) depend on the content overlap or statistical properties, such as data range and pattern, to determine the similarity of two attributes.

Recent works ([16], [17], and [21]) on matching query interfaces of Web databases fall into the first category, based on identifying the attribute labels by examining the descriptive text surrounding interface elements and finding *synonym relationships* between the labels. Such methods are not stable and robust in the Web context as no description may exist, or the identified description may not be informative. On the other hand, instance-based schema matching was seldom employed in the deep Web scenario because of the difficulty of automatically acquiring database contents hidden behind search interfaces. Paradoxically, a key prerequisite for automatic data acquisition from the deep Web is to understand the semantics of search elements.

Different from the previous work, our goal is to understand and match not only interface schemas but also result schemas of Web databases. Consequently, the label-based matching approach is insufficient and even inapplicable due to the frequent lack of explicit attribute labels and descriptions in result pages. Therefore, we propose an instance-based solution to this problem. We first submit semantically pre-identified query terms through search interfaces (section 3.2). After obtaining returned result data, we then analyze the results to understand the semantics of both the query interfaces and data attributes, as well as to match the homogeneous schemas of different Web databases (section 4).

3.1 Observation

During the interaction with Web databases, we observe two interesting phenomena.

On the one hand, when an improper query is submitted to a Web database there are often few results or even no results returned. Improperness here means the query keywords submitted into a particular element are not applicable values of the database attribute to which the element is associated. Taking the Web database shown in Figure 1 as an example, the site reports only 4 matches for the query “Harry Potter” when submitted into the “Author” element, while it reports 228 matches for the same query when submitted into the “Title” element. On the other hand, we observe that when a proper query that returns a result web page is submitted into the input elements of a Web database, then the query keywords very likely reappear in the returned result’s corresponding attributes. For example, in Figure 1, when we submit query “Harry Potter” into the “Title” element, the three returned book instances all contain the query keywords (i.e., Harry Potter) in their “Title” attribute.

Generally, how many times the keywords for a query re-appear in the result pages and where they appear tell us important information about both the interface schema and the result schema. Specifically, if we employ the keywords or values belonging to some semantically pre-identified data attributes as queries to submit into a Web database, we can accomplish two tasks. First, the re-occurrence of the query keywords in the returned results can be used as an indicator of which query submission is appropriate (i.e., the data values are submitted into the semantically associated elements in order to discover the query interface schema). Second, the position or location of the submitted query keywords in the result pages can be used to identify the semantically associated attributes (i.e., to discover the result schema).

3.2 Query Probing

Given some target Web databases in a specific domain, our query probing process aims to send domain-specific queries to these target Web databases and collect their returned results for later analysis.

To accomplish the task, we make two assumptions about the query probing process. First, a global schema for the specific domain is pre-defined or pre-generated. Second, a number of sample data objects under the domain global schema are also available. In fact, global schema generation over information sources to conceptualize the underlying domain is an interesting problem. Proposed approaches rely on either the names of the schema elements and the structure of the schema ([7] and [16]) or formal ontologies ([4] and [15]). We consider this problem as a separate research direction, and do not deal with it in this paper. In our experiments, we manually define the global schema and collect sample instances. In future work, we plan to implement one of the previously proposed approaches to automatically generate a global schema over a sample set of Web databases and then map new Web databases to the generated global schema.

3.2.1 Workflow

We show in Figure 3 the workflow of an automatic probing process. Given the Web database with its search interface, an element identification component first locates qualified input elements in the search interface. Equipped with instances under a global schema, a query submission component then exhaustively submits the attribute values of pre-known instances into those identified input elements. After collecting the returned results for all submitted queries, a wrapper induction component induces a regular-expression wrapper composed of HTML-tags. Next, a data extraction component employs the induced wrapper to extract structured data objects from query result pages and arrange them into a data table. Finally, the re-occurrences of submitted queries in the columns of this table are counted and stored into a *query occurrence cube*, which will be introduced in the next subsection.

Given a Web database, the first task is to identify input elements in its search interface and it can be done by searching for the input-related tags[6] in a HTML searchable form. In the HTTP protocol, a query submission is done by sending a query request to the server containing the names of input elements and their corresponding query terms. In this work, we consider to submitting one value to one element each time while keeping the default values of other elements. For each TEXTBOX element in HTML forms, as we do not know its value domain, we exhaustively try all the attribute values from the given sample instances. For each SELECT element, its domain values are limited to its OPTION

[6] Please refer to the HTML specification [24].

elements (i.e., we can only choose one or more of its OPTIONs as the query terms). Thus, for each attribute value of the given instances, we try to find and submit an option "similar"[7] to the value. For other elements like CHECKBOX and RADIOBOX, the process is similar. As a consequence, the maximum submission time will be the product of the number of attributes in the global schema, the number of provided sample instances and the number of interface elements considered.

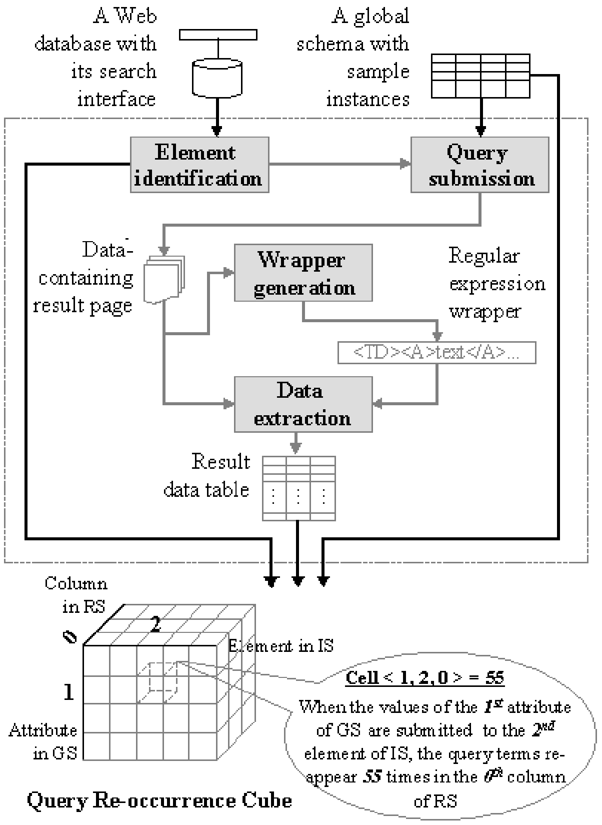

A Web database with its search interface
A global schema with sample instances
Element identification
Query submission
Data-containing result page
Wrapper generation
Regular expression wrapper
<TD><A>text</A>...
Data extraction
Result data table
Column in RS
0
2
Element in IS
1
Attribute in GS
Cell < 1, 2, 0 > = 55
When the values of the 1st attribute of GS are submitted to the 2nd element of IS, the query terms re-appear 55 times in the 0th column of RS
Query Re-occurrence Cube

Figure 3. Flow of the query probing process and the query re-occurrence cube.

After sending queries to the identified input elements and collecting returned result pages from the Web database, the next task is to extract structured data from the pages. While dealing with hundreds or possibly thousands of Web databases in one domain, each of which encloses its data in the result pages according to some specific HTML-tag structures, how to automatically extract data objects from the pages is a very challenging problem and it has attracted increasing research interest. Recently, attempts were made to develop fully automatic approaches for inducing wrappers to extract embedded semi-structured data content from dynamic template-generated HTML pages ([1], [9], [10] and [23]). Discussion of these approaches is beyond the scope of this paper and interested readers are referred to the above papers for further information.

In this paper, we choose our previous work [23] to induce a regular-expression wrapper based on nested repeated-pattern discovery in HTML pages. We also employ the data extraction module of [23] to extract the enclosed data objects into a table so that each column of the result table corresponds to one attribute of the returned result. That is, each column of the result table is one observed attribute of the result schema.

3.2.2 Query Occurrence Cub

After counting the re-appearance of each submitted value in the query results, a *Query Occurrence Cube (QOCube)* is constructed for the target Web database, as shown in Figure 3. The cube height represents the number of attributes in the pre-known global schema. The cube width represents the number of interface elements considered (i.e., attributes of interface schema). The cube depth is the number of columns in the result table (i.e., attributes of result schema). Moreover, each cell in this cube stores an occurrence count associated with the three dimensions. For example, in Figure 3, *cell<1, 2, 0>* equal to *55* means that when the values for the *1st* attribute of GS are submitted to the *2nd* element of IS the query terms re-appear *55* times in the *0th* column of RS.

Conveniently, the constructed *QOCube* provides a unified solution to match the 3 pairs of Web database schemas. The 3-dimensional cube can be easily projected onto three *Query Occurrence Matrices* (front, top and left), which exactly reflect the relationship between pairs of the three schemas (i.e., IS with GS, IS with RS, and GS with RS). Suppose the number of attributes in the global schema is *N*, the number of elements in the interface schema is *M*, and the number of columns in the result table is *L*. Once a project function is selected, say *sum*, the 3-dimensional cube $QOC_{N\times M\times L}$ can be projected into three 2-dimensional *occurrence matrices*, $OM^{IG}_{M\times N}$ for IS and GS, $OMat^{RG}_{L\times N}$ for GS and RS, and $OM^{IR}_{M\times L}$ for IS and RS. The main research issue now becomes how to find the correspondence between a pair of schemas in the projection matrices.

4. Instance-based Schema Matching

4.1 Intra-site Schema Matching

In this section, we focus on how to match the attributes between IS and GS, RS and GS, and IS and RS based on the obtained matrices: $OM^{IG}_{M\times N}$, $OM^{RG}_{L\times N}$, and $OM^{IR}_{M\times L}$.

An example[8] of $OM^{IG}_{5\times 4}$ is shown in Example 1 with the correct matching in the gray rectangles, when GS = {Title_{GS}, Author_{GS}, Publisher_{GS}, ISBN_{GS}} and IS = {Author_{IS}, Title_{IS}, Publisher_{IS}, Keyword_{IS}, ISBN_{IS}}.

[7] The attribute value and the option value (two text strings) are similar as long as they contain at least one common keyword.

[8] All examples in this section are obtained from real Web databases in our experiments.

EXAMPLE 1:

$$\begin{array}{r|cccc|l} & \text{Title}_{GS} & \text{Author}_{GS} & \text{Publisher}_{GS} & \text{ISBN}_{GS} & \\ \text{Author}_{IS} & 93 & \boxed{498} & 534 & 0 & \\ \text{Title}_{IS} & \boxed{451} & 345 & 501 & 0 & \\ \text{Publisher}_{IS} & 62 & 184 & \boxed{468} & 2 & \\ \text{Keyword}_{IS} & 120 & 248 & 143 & \boxed{275} & \\ \text{ISBN}_{IS} & 0 & 0 & 0 & 258 & _{5\times 4} \end{array}$$

In fact, there are some properties of the occurrence matrix to consider when searching for the correspondence or correlation between the rows and columns of the matrix representing the attributes of the two schemas. First, an absolute high occurrence may not represent a correct matching. For example, the matrix element for Author_{IS} and Publisher_{GS} (534) is the highest value in the matrix while Author_{IS} and Publisher_{GS} do not semantically correspond to each other. Second, given a particular matrix element m_{ij}, its relative magnitude among all elements for its row i and column j is more important than its absolute value. For example, for Keyword_{IS}, which is in fact not a real attribute for Book objects, its similar performance on all columns indicates that it may not be a good match for any one of the columns. The element of Publisher_{IS} and Publisher_{GS} (468) is not the highest one among the elements for Publisher_{GS}. However, it is relatively larger than the other elements that for Publisher_{IS} and the other attributes of GS.

Interestingly, we can regard the problem as follows. By sending sample queries, a part of the database content relevant to the queries is fetched from the primary backend database. For any two schemas, S_1 and S_2, of the Web database, the obtained database content can be partitioned according to the attributes of S_1 and S_2, respectively. Suppose the partitions by the attributes of S_1 are $A_1, A_2, \ldots A_n$ and the partitions by the attributes of S_2 are $B_1, B_2, \ldots B_m$. The element m_{ij} in the occurrence matrix for S_1 and S_2 actually indicates the content overlap between partitions A_i and B_j with respect to the occurrences of submitted query values in the partitions. The schema matching problem now becomes finding the pair of partitions that belong to two schemas (e.g., IS and GS) such that their overlap with each other is more than their overlap with other partitions belonging to the opposite schema.

To solve this problem, a *mutual information* concept, also known as *cross-entropy* and *information gain* in Information Entropy theory, can be introduced as it interprets the overlap between two partitions X and Y of a random event set as the "information about X contained in Y" or the "information about B contained in X" ([20]).

DEFINITION 1: Suppose X and Y are two partitions over a collection of events, and x_i and y_j are partition elements of X and Y with joint probability $p(x_i, y_j)$ and respective marginal probability $p(x_i)$ and $p(y_j)$. The *mutual information* of the partition A and B is

$$I(X;Y) = \sum_i \sum_j p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$$

Accordingly, we can estimate the mutual information between a pair of attributes from two schemas using the following definition.

DEFINITION 2: Given a query occurrence matrix $OM^{S_1S_2}{}_{I\times J}$ of two schemas S_1 and S_2, the *estimated mutual information (**EMI**)* between the i^{th} attribute of S_1 (say A_i) and the j^{th} attribute of S_2 (say B_j) is

$$EMI(A_i, B_j) = \frac{m_{ij}}{M} \log \frac{\frac{m_{ij}}{M}}{\frac{m_{i+}}{M} * \frac{m_{+j}}{M}}$$

with M being $\sum_{i,j} m_{ij}$, m_{i+} being $\sum_i m_{ij}$ and m_{+j} being $\sum_j m_{ij}$.

Thus, the occurrence matrix in Example 1 can induce the *EMI* matrix shown in Example 2, with each matrix element being the estimated mutual information value for the corresponding schema attributes.

EXAMPLE 2:

$$\begin{array}{r|cccc|l} & \text{Title}_{GS} & \text{Author}_{GS} & \text{Publisher}_{GS} & \text{ISBN}_{GS} & \\ \text{Author}_{IS} & -0.04 & \boxed{0.11} & 0.06 & 0.00 & \\ \text{Title}_{IS} & \boxed{0.19} & -0.03 & -0.01 & 0.00 & \\ \text{Publisher}_{IS} & -0.03 & -0.02 & \boxed{0.14} & -0.01 & \\ \text{Keyword}_{IS} & -0.01 & 0.01 & -0.07 & 0.17 & \\ \text{ISBN}_{IS} & 0.00 & 0.00 & 0.00 & \boxed{0.32} & _{5\times 4} \end{array}$$

To find a 1-1 attribute matching of the two schemas is easy in the *EMI* matrix. If one matrix element is larger than the other elements in the same row and also larger than the other elements in the same column, its represented attributes will have a larger overlap between each other than their overlap with other attributes of the opposite schema, as shown by the gray rectangles. For example, *EMI*(Author_{IS}, Author_{GS}) = 0.11 is the largest one in both its row and its column, and it is a correct match. Therefore, we propose the following definition to quantify the intra-site schema matching.

DEFINITION 3: Assume two schemas S_1 and S_2 with the corresponding *EMI* matrix $[e_{ij}]$. The i^{th} attribute of S_1 *matches* with the j^{th} attribute of S_2 if $e_{ij} \geq e_{ik} \mid k \neq j$ and $e_{ij} \geq e_{kj} \mid k \neq i$.

4.2 Inter-site Schema Matching

In this section, we focus on how to find the corresponding attributes for homogeneous schemas of different Web databases, namely, IS and IS, and RS and RS.

We propose an approach to match interface/result schemas of different Web databases from scratch by computing *vector similarity* "borrowed" from the *Vector*

Space Model in Information Retrieval [2]. In vector space model, documents are represented as vectors in a multi-dimensional space. In the space, each dimension represents a term or concept found in a document and the values are the corresponding frequencies of the terms in the document. Similarity between two vectors is measured by the cosine of the angle between their two vectors, which is computed as the inner product of the two vectors, normalized by the products of the vector lengths.

Inspired by the above idea, if we consider each attribute of an individual interface/result schema as a "document" and each attribute of the global schema as a "concept"; then each row in the occurrence matrix represents a corresponding document vector in the space. Therefore, we can calculate the similarity (i.e., semantic correspondence) between attributes from different schemas by measuring their vector similarity. We give the following definition to quantify the inter-site schema matching between two Web databases.

DEFINITION 4: Given two query occurrence matrices of two Web databases' interface/result schemas $OM^{S_1G} = [a_{ij}]_{n\times m}$ and $OM^{S_2G} = [b_{ij}]_{l\times m}$ with respect to the same global schema, the *estimated vector similarity (**EVS**)* between the i^{th} attribute of S_1 (say A_i) and the j^{th} attribute of S_2 (say B_j) is

$$EVS(A_i, B_j) = \frac{\sum_k a_{ik} b_{jk}}{\sqrt{\sum_k a_{ik}^2} * \sqrt{\sum_k b_{jk}^2}}$$

EXAMPLE 3:

$$\begin{array}{c|cccc} & T_G & A_G & P_G & I_G \\ \hline A_1 & 93 & 498 & 534 & 0 \\ T_1 & 451 & 345 & 501 & 0 \\ P_1 & 62 & 184 & 468 & 2 \\ K_1 & 120 & 248 & 143 & 275 \\ I_1 & 0 & 0 & 0 & 258 \end{array}_{5\times 4}$$

Occurrence Matrix of S_1

$$\begin{array}{c|cccc} & T_G & A_G & P_G & I_G \\ \hline T_2 & 166 & 177 & 118 & 0 \\ A_2(P) & 39 & 331 & 406 & 0 \\ I_2 & 0 & 0 & 0 & 18 \end{array}_{3\times 4}$$

Occurrence Matrix Of S_2

$$\begin{array}{c|ccc} & T_2 & A_2(P) & I_2 \\ \hline A_1 & 0.84 & \boxed{0.99} & 0 \\ T_1 & \boxed{0.96} & 0.84 & 0 \\ P_1 & 0.71 & 0.95 & 0.01 \\ K_1 & 0.72 & 0.67 & 0.66 \\ I_1 & 0 & 0 & \boxed{1.00} \end{array}_{5\times 3}$$

Vector Similarity Matrix

To find a 1-1 attribute matching of two schemas in the *EVS* matrix is the same as in the *EMI* matrix (Definition 3). A matrix element that is the largest both in its row and column represents a matching. For instance, Example 4 shows two occurrence matrices of two interface schemas with respect to a global schema GS = {Title, Author, Publisher, ISBN}, where IS_1 = {$Author_1$, $Title_1$, $Publisher_1$, $Keyword_1$, $ISBN_1$}, IS_2 = {$Title_2$, $Author_2$, $ISBN_2$}. Grey rectangles depict the chosen largest similarity values among rows and columns, which also show the correct matching. Interestingly, although the second attribute of IS_2, $Author_2$, is somehow wrongly matched to $Publisher_2$ of GS in the previous intra-site schema matching, the method still can find the right mapping.

4.3 Cross Validation

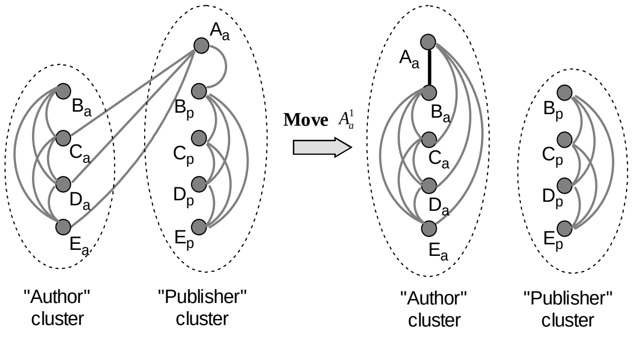
Given multiple Web databases in the same domain, we can employ the techniques proposed in sections 4.1and 4.2 to identify the matching attributes belonging to schemas of an individual Web database and the matching attributes belonging to schemas of different Web databases. As a consequence, we can employ the five categories of matching results (i.e., GS-IS, GS-RS, IS-RS, IS-IS and RS-RS) to cross validate each other (i.e., to recognize which matching is correct and which is not). In this section, we focus on how to cross validate different matching results produced from both "inter-site" and "intra-site" matching. Note that in this step, we do not limit how the schemas are previously matched (i.e. we can employ any applicable label-based or instance-based method) as long as the matching results are provided.

Given all the attributes from the interface schemas (or result schemas) of the Web databases, we can categorize the IS (or RS) attributes into multiple clusters with respect to GS attributes to which they have been matched. For example, the attributes are previously matched to the attribute A_G of the global schema is categorized into one cluster while the attributes are previously matched to the attribute T_G of the global schema is categorized into another cluster. Recall that attributes are also matched to each other in inter-site schema matching. In the perfect case, an attribute in one cluster only matches with attributes in the same cluster. When a mapping across clusters does exist (i.e., two attributes in two different clusters have a match) there must be a mismatch. The possible reason for the mismatch could be either that one of the two attributes was put into the wrong cluster or that the matching between these two attributes is wrong.

Interestingly, if we consider the attributes as vertices and matching between attributes as edges, we can convert the problem of deciding which matchings are incorrect into a *graph partitioning problem*: given a set of vertices and edges, divide the vertices into *N* partitions such that the edge-cut is minimized. The *edge-cut* is the sum of the weights (1 in this case) of all the edges between the partitions. This *graph partitioning problem* is known to be NP-hard [11]. Therefore, we can only expect approximate solutions in general.

In our case, where there is already an initial partition of the vertices (according to the matching results with respect to GS), a simple approximate approach is to move vertices over partitions as long as the number of cuts would decreases; a vertex *v* is moved to the partition in which most of its "neighbours" reside. Since a vertex *v* needs to be moved if many of its neighbours jump, multiple passes are likely to be needed before the process converges on a local optimum. When the process stops, we resolve the cross cluster matching between attributes A_i of site S_1 and B_j of site S_2 contained in two clusters C_1 and C_2 by discarding it and re-matching A_i to attribute B_k of site S_2 clustered into C_1 or vice versa.

EXAMPLE 4:



Example 4 illustrates one pass of such an approximate approach. For simplicity, suppose that the global schema only contains two attributes {Author, Publisher} and there are five Web databases with the IS attributes $IS_1 = \{A_a\}$, $IS_2 = \{B_a, B_p\}$, $IS_3 = \{C_a, C_p\}$, $IS_4 = \{D_a, D_p\}$ and $IS_5 = \{E_a, E_p\}$. The two ellipses in the left depict how the attributes are primarily clustered according to which GS attribute they are matched to (by intra-site schema matching), and the edges between two attributes show whether they are matched or not (by inter-site schema matching). In the initial state, A_a is wrongly matched to the Publisher attribute of GS and also wrongly matched to B_p while it has been correctly matched to three other attributes in the Author cluster. Therefore, A_a is moved to decrease the number of edges across clusters from 3 to 1, as shown in Example 4. By such a "moving" process, we correct the matching attribute of A_a from the Publisher to the Author attribute of GS. After the move, the edge between A_a and B_p is replaced by a new edge between A_a and B_a (the attribute of site 2 that is matched to the global attribute Author).

Due to space limitations, we omit the detailed algorithm for the above cross-validation technique and only show the experimental results in the next section to verify its effectiveness.

5. Experiments

We performed a comprehensive evaluation of the proposed instance-based schema matching approaches on thirty complex Web databases over two domains: Book and Used Car. The main goal was to investigate the feasibility of a unified and accurate solution to matching schemas both in a single site and from different sites. We first describe the Web databases employed for the testing. Then we present the results for intra-site schema matching and inter-site schema matching, and the improvement achieved by cross validating the matching results.

5.1 Test Web Databases

For our evaluation, we used 20 Web databases for purchasing books online and 10 Web databases for searching for used cars online. The global schema for the two domains are manually defined as Book = {Title, Author, Publisher, ISBN} and UsedCar = {Make, Model, PostalZip, State, Price, Mileage, Year}. We also manually collected 20 book instances and 10 car instances (details are shown in Table 5 and Table 6 in the Appendix) and took their attribute values as sample queries to be used to probe the test Web databases. After obtaining the query result pages from each Web database, we employed the previous work [23] on wrapper induction to automatically extract the result records according to their specific structures and re-arrange them into a result table.

Table 1. Characteristics of test Web databases.

	#Interface Element	#TS	%SS	#Result Column	#Extracted Data
Book	4.2	343.3	32%	6.25	1322.9
UCar	6.0	123.1	72%	5	995.3

The columns #TS and %SS of Table 1 represent, respectively, for the number of total submissions made to the test Web databases and the corresponding success rate[9]. The reason that the UsedCar domain has a lower number of submissions and a higher success rate than the Book domain is because, SELECT and TEXTBOX are treated differently when submitting the queries. We exhaustively tried all the attributes of the pre-known instances for a TEXTBOX element, while we only submit the OPTION values of a SELECT element if they are found to be similar to one or more attribute values of the pre-known instances (see section 3.2.1). In our experiments, most of the Web databases in the Book domain only contain TEXTBOX elements. Therefore, it has a higher number of submission and lower success rate.

5.2 Matching Results

5.2.1 Intra-site and inter-site

In this subsection, we report and discuss the experimental results for both intra-site and inter-site schema matching of the two domains. The intra-site schema matching results are listed in Table 2. To verify the effectiveness of our proposed instance-based matching approach (EMI)

[9] A query submission is successful if the induced wrapper can extract at least one instance from the query result page.

derived from mutual information analysis, we implemented a simple method as our baseline (MAX). The baseline method works as follows: in the query value occurrence matrix, the matrix element with the largest value both among the elements in the same column and among the elements in the same row is identified as an attribute matching.

In our evaluation, precision and recall originating from the information retrieval area are borrowed as the metrics. Precision here is measured as the ratio of the number of correctly identified matching attribute-pairs to the total number of attribute-pairs identified by the methods. Recall here is measured as the ratio of the number of correctly identified matching attribute-pairs to the total number of matching pairs in the two schemas. Suppose the number of correctly identified matching attribute-pairs is *C*, the number of wrongly identified matching attribute-pairs is *W* and the number of correct matching attribute-pairs but somehow missed in the approach is *M*, then the precision of the approach is $\frac{C}{C+W}$ and its recall is $\frac{C}{C+M}$.

Table 2. Intra-site schema matching results.

		IS – GS		RS – GS		IS – RS	
		P	**R**	**P**	**R**	**P**	**R**
Book	MAX	68%	50%	91%	81%	90%	84%
	EMI	80%	71%	95%	88%	93%	87%
Car	MAX	97%	63%	96%	57%	100%	67%
	EMI	97%	64%	93%	63%	100%	73%

In Table 2, we can see that our *EMI-based* method significantly outperforms the baseline method, especially for global-interface schema matching. In the Book domain, both the *EMI-based* and *Max-based* methods produce the worst results on IS-GS schema matching. The reason is that Web databases of this domain tend to include a Keyword input element in the interface schema for the convenience of end-users who may want to use keyword search. Using the Keyword element often returns results for any query no matter to which global attribute the query belongs. Since there is not a noisy keyword attribute in the global schemas and the result schemas, our matching approach can achieve a higher accuracy in GS-RS matching. In the UsedCar domain, both MAX-based and EMI-based methods have a relatively low recall. The reason is that our matching techniques are based on counting the re-appearances of submitted queries in the result data, which is more suitable for database attributes accepting the "equal" select operator. When handling numeric-field attributes that accept "less than" or "greater than" select operators, such as Price and Mileage, the returned results sometimes may not include the exact query term, such as "$10,000".

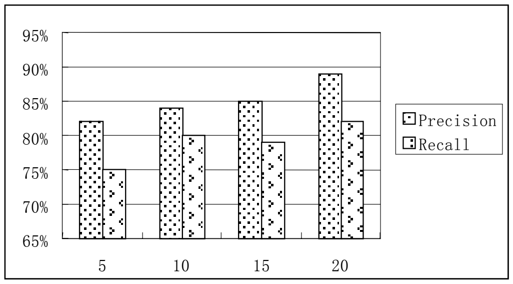


Figure 4. Result achieved by different number of sample instances.

We show in Figure 4 how the achieved results vary when the number of sample instances is increased. Columns in Figure 4 are achieved average precision and recall of the intra-site schema matching results of the Book domain, when the number of instances is set to 5, 10, 15 and 20. From the figure, we can see that the achieved results generally increase when the number of sample instances increase. However, more sample instances mean more query submissions to the Web server. Since we do not want to overburden the target Web databases, an interesting future research direction might be to find a trade-off between the number of submission and the achieved results.

Table 3. Inter-site schema matching results.

		EVS		Label-based	
		P	**R**	**P**	**R**
Book	IS – IS	80%	71%	90%	87%
	RS – RS	94%	86%	95%	14%
Car	IS – IS	92%	72%	89%	88%
	RS – RS	89%	66%	98%	25%

In Table 3 we compare the inter-site schema matching results achieved by our proposed approach (*EVS*) based on *vector similarity* analysis with label-based approaches. Label-based approaches are mainly based on finding the synonym relationship between attribute labels. In matching interface schemas, we employ previous approaches on identifying the surrounding text of input elements as their labels ([16], [17], and [21]). In matching result schemas, we try to find either explicit author-supplied column headers in the result pages or the text strings commonly shared by all extracted instances as their attribute labels. Table 3 shows that the performance of the *EVS-based* method is close to that of label-based methods in IS-IS matching, while it performs much better in RS-RS matching when attribute labels are often unavailable in result pages. In addition, our approach does not require intelligent layout analysis to precisely identify the right attribute labels.

5.2.2 Cross Validation

We present in Table 4 the effectiveness of the proposed cross validation approach in improving the overall accuracy. The approach is based on an approximate solution of the graph partitioning problem.

Table 4. Effectiveness of cross validation.

		Before CV		After CV	
		P	**R**	**P**	**R**
Book	IS – GS	80%	71%	96%	83%
	RS – GS	95%	88%	98%	91%
	IS – RS	94%	88%	97%	90%
	IS – IS	91%	70%	94%	74%
	RS – RS	94%	86%	99%	87%
Car	IS – GS	97%	64%	97%	72%
	RS – GS	93%	63%	97%	70%
	IS – RS	100%	73%	100%	75%
	IS – IS	92%	72%	95%	77%
	RS – RS	89%	66%	92%	69%

Table 4 shows that the cross validation method does improve the overall matching accuracy, especially in the Book domain. It is notable that we cannot achieve a recall as high as we can achieve the precision (over 90%). The cause is not due to the ineffectiveness for the cross validation but may be due to the probing-based approach itself. In our experiments, we observe some issues that need further considerations.

The performance of our instance-based matching approaches to some extent depends on the selection of the sample instances. More specifically, two properties of the sample instances could influence the matching process, the topics that the sample instances cover and the attribute-distinguish capability of the sample instances. Take the Book domain as an example. Some Web databases may only contain books about computer programming while others only have novels. Therefore, to ensure that useful instances can be extracted from the Web databases' answers to the sample queries, various topics are required to be covered in the sample instances. At the same time, the attribute-distinguish capability of the sample instances may also influence the matching results. For example, the name of a famous person usually frequently appears both in the Author attribute of the books he/she wrote and the Title attribute of his/her biographies, such as "Jane Austen" in our chosen sample instances.

We also notice that, as Web databases vary in their designs, some of them might generate result pages with different formats for different queries. For example, when answering a Title query, a Web database returns a list of qualified book instances and each of the instances is described by come text. However, when answering an ISBN query, the same Web database returns only one unique book instance with its detailed information shown in the result page. It is obvious that these two kinds of results are generated by two different templates. To deal with this issue, an intelligent result analysis method is needed to first extract results with different formats and then combine them into one uniform result table.

6. Related Work

Schema matching is a basic problem in database research with numerous techniques proposed to address the problem (see [12] and [22] for surveys). Existing work that addresses the problem of automatic schema matching for deep Web sources adopts the prior techniques on matching schemas of traditional databases. [16] presented a statistical approach to integrate the query interface schemas of deep websites in the same domain. It hypothesizes that given deep-web sources in the same domain, the aggregate vocabulary describing the interface input elements tends to have a relatively small size. Furthermore, there exists a unified hidden schema underlying these interfaces. A statistical probability model is employed to find the hidden schema by the co-appearance of attribute names. The schema matching methods employed in this paper are label-based. WISE-Integrator, a tool that performs automatic integration of Web search interfaces is presented in [17]. WISE-Integrator employs sophisticated techniques to identify matching attributes from different search interfaces for integration. This can also be classified as a label-based method since it mainly relies on the approximate string match between attribute names. [18] investigated algorithms for generic schema matching, outside of any particular data model or application. An algorithm called Cupid was proposed to discover mappings between schema elements based on their names, data types, constraints, and schema structure. [19] used a classifier to categorize attributes according to their field specifications and data values, and then train a neural network to recognize similar attributes. However, this method may not be applicable for Web databases since both field specifications and data values are incomplete in many cases. [12] developed the COMA schema matching system as a platform to combine multiple matchers in a flexible way. While their approach may seem similar to our cross validation method, it is fundamentally different since the goal of our method is the reinforcement of multiple matchers, not the straightforward combination of the matchers. [21] presented HiWe, a prototype deep-web crawler that can extract the labels of interface elements and automatically submit queries through the elements. In this work, interface elements with the same/similar labels are matched in order to obtain each other's domain values for automatic query submission.

The main difference between our work and previous works is that we aim to provide a general framework for schema matching of Web databases. To the best of our knowledge, no previous work has presented such a framework, especially the combined schema model for deep websites. Moreover, the instance-based schema matching method is seldom used for schema matching in the deep Web context since it is hard to get instances from hidden databases. Supplied with a set of sample instances, our work proves that instance-based methods are very effective for Web database schema matching.

7. Conclusion

In this paper, we investigate the problem of schema matching for Web databases. We propose a combined schema model to describe various schemas in a deep website and a generative view to include five kinds of schema matching of related Web databases in a specific domain.

In the combined schema model, we address two significant schema matching problems for Web databases, intra-site schema matching and inter-site schema matching. The first problem is crucially important in automatically extracting data from Web databases, while the second problem is of significant importance in integrating data from different Web databases. In the generative view, we then investigate a unified solution to the two problems based on domain-specific query probing and attribute content overlap. Our instance-based approaches, specifically adopting the mutual information concept and vector similarity analysis, are quite powerful for precisely identifying the matching relationships among attributes for intra-site and inter-site matching. Benefiting from our general framework, a cross validation technique, converted to a graph partitioning problem, is introduced and shown to improve on the matching performance.

As mentioned, in current stage our approach needs some human involvement to provide a precise global schema and instance samples. One direction to extend this work is to adopt automatic global schema generation techniques to make the whole system fully automatic. Another direction of improvement is to combine our works with previous label-based approaches to build a more robust matching system that can handle most attributes of Web databases.

References

[1] A. Arasu, and H. Garcia-Molina. *Extracting structured data from Web pages*. Proc. ACM SIGMOD Conf., 2003.

[2] R. Baeza-Yates, and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.

[3] C. Batini, M. Lenzerini, and S. B. Navathe. *A comparative analysis of methodologies for database schema integration*. ACM Computing Surveys, **18**(4), 323-364, 1986.

[4] D. Beneventano, S, Bergamaschi, F. Guerra and M. Vincini. *Synthesizing an integrated ontology*. Internet Computing, vol 7, no. 5, 2003.

[5] BrightPlanet Corp. *The deep web: surfacing hidden value.* http://www.completeplanet.com/Tutorials/DeepWeb/

[6] J. Callan, M. Connell and A. Du. *Automatic discovery of language models for text databases.* Proc. ACM SIGMOD Conf., 1999.

[7] S. Castano, V. Antonellis, and S. Vimercati. *Global viewing of heterogeneous data sources*. IEEE Trans. Data and Knowledge Eng., vol 13, no. 2, 2001.

[8] C.H. Chang, B. He, C. Li, and Z. Zhang: *Structured Databases on the Web: Observations and Implications discovery*. Technical Report UIUCCDCS-R-2003-2321. CS Department, University of Illinois at Urbana-Champaign. February, 2003.

[9] C.H. Chang, and S.C. Lui. *IEPAD: information extraction based on pattern discovery*. Proc. 10th World Wide Web Conf., 681-688, 2001.

[10] V. Crescenzi, G. Mecca and P. Merialdo. *ROADRUNNER: towards automatic data extraction from large web sites*. Proc. 27th VLDB. Conf., 109-118, 2001.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[12] H. Do and E. Rahm. *COMA: a system for flexible combination of schema matching approaches*. Proc. 28th VLDB Conf., 2002.

[13] A. Doan, P. Domingos and A. Halevy. *Reconciling schemas of disparate data sources: a machine-learning approach.* Proc. ACM SIGMOD, 2001.

[14] D. Florescu, A.Y. Levy, and A.O. Mendelzon. *Database techniques for the world-wide web: a survey*. SIGMOD Record **27**(3), 59-74, 1998.

[15] F. Hakimpour, and A. Geppert. *Global schema generation ssing formal ontologies*. Proc. 21st Conf. on Conceptual Modeling, 2002.

[16] B. He, and C.C. Chang. *Statistical schema matching across Web query interfaces*. Proc. ACM SIGMOD Conf., 2003.

[17] H. He, W. Meng, C. Yu and Z. Wu. *WISE-Integrator: an automatic integrator of Web search interfaces for E-commerce*. Proc. 29th VLDB Conf., 2003.

[18] J. Madhavan, P.A. Bernstsein and E. Rahm. *Generic schema matching with Cupid*. Proc. 27th VLDB Conf., 2001.

[19] W. Li and C. Clifton. *Semantic integration in heterogeneous databases using neural networks.* Proc 20th VLDB Conf, 1994.

[20] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 1984.

[21] S. Raghavan and H. Garcia-Molina. *Crawling the hidden web*. Proc. 27th VLDB Conf., 129-138, 2001.

[22] E. Rahm and P.A. Bernstein. *A survey of approaches to automatic schema matching*. VLDB Journal, **10**(4), 334-350, 2001.

[23] J. Wang and F. Lochovsky. *Data extraction and label assignment for web databases*. Proc. 12th World Wide Web Conf., 187-196, 2003.

[24] World Wide Web Consortium. *HTML 4.01 Specification*, 1999.

APPENDIX

Table 5. Test Web databases.

Homepage URL	#Interface elements	#Submission	#Successful submission	Success rate	#Result columns	#Extracted instances
www.bookstore.co.uk	3	240	55	23%	3	326
dogbert.abebooks.com	7	560	241	43%	8	4014
www.powells.com	6	480	156	33%	7	2941
www.randomhouse.com	5	400	133	33%	8	3162
www.hwg.org/bookstore	4	320	124	39%	6	778
www.booksandcollectibles.com.au	4	320	139	43%	8	1519
www.bestbookdeal.com	3	240	181	75%	4	1436
search.barnesandnoble.com	3	240	98	41%	7	394
www.mysimon.com	4	320	125	39%	4	3205
isbn.nu/advanced.html	5	400	65	16%	6	2715
www.a1books.com	5	400	63	16%	5	1034
www.booksinc.net	2	160	80	50%	5	1869
www.page1book.com	5	400	197	49%	12	1746
www.biggerbooks.com	4	320	174	54%	7	1498
www.bookcloseouts.com	3	240	58	24%	7	710
www.christianbook.com	5	400	82	21%	10	549
www.hamiltonbook.com	5	400	47	12%	5	189
www.textbookx.com	3	240	98	41%	6	1069
www.1bookstreet.com	6	480	104	22%	6	866
www.allbookstores.com	5	400	72	18%	6	1384
Average	4.2	343.3	98.2	32%	6.25	1322.9

(a) Book Domain.

Homepage URL	#Interface elements	#Submission	#Successful submission	Success rate	#Result columns	#Extracted instances
www.consumerreports.org	5	25	23	92%	2	213
www.bigbillybarrett.com	5	33	32	97%	4	355
www.carbuyer.com	2	21	20	95%	7	500
www.401carfinder.com	7	359	133	37%	4	1104
www.2buycars.net	14	381	230	60%	9	2334
auto.consumerguide.com	5	97	34	35%	5	501
www.fredbondesen.com	4	26	23	88%	5	231
midland.autochooser.com	9	169	167	99%	2	4355
www.mswebmasters.com	4	21	20	95%	5	203
toyota.traversemotors.com	5	99	25	25%	7	157
Average	6	123.1	70.7	72.4%	5	995.3

(b) Used Car Domain.

Table 6. Instances for sample query probing.

Title	Author	Publisher	ISBN
Story of Art	Gombrich	Phaidon Press	0714832472
Winning Through Intimidation	Ringer	Fawcett Books	0449207862
New Joy of Cooking	Irma Ruer	Scribner	0684818701
Worst Case Scenario Survival Handbook	Joshua Piven	Chronicle Books	0811825558
Fight Cancer Win	Fischer	Agora Health	1891434012
Interpretation of Dreams	Sigmund Freud	Avon Books	0380010003
Sense and Sensibility	Jane Austen	Norton Company	039397751X
Guess How Much I Love You	Sam McBratney	Candlewick Press	076360013X
On the Shoulders of Giants	Stephen Hawking	Running Press	0762413484
Selfish Gene	Richard Dawkins	Oxford University Press	0192860925
Communion With God	Walsch	Berkley Pub Group	0399146709
NBA Alphabet	Mayers	Abrams Books	0810931435
Core Java	Gary Cornell	Prentice Hall	0130471771
Art of War	Sun Tzu	Delacorte Press	0385292163
Shining	Stephen King	Pocket Books	0743424425
Harry Potter	Rowling	Scholastic	0439139597
Lord of the Rings	Tolkien	Houghton Mifflin	0618260587
Programming Language	Bjarne Stroustrup	Addison Wesley	0201700735
Who Moved My Cheese	Spencer Johnson	Putnam Pub Group	0399144463
Making of the Microsoft Empire	James Wallace	HarperBusiness	0887306292

(a) Book Domain.

Make	Model	Postal	State	Price	Mileage	Year
Acura	TL	10021	New York	$1,000	10,000	1993
Audi	A4	20854	Maryland	$2,000	20,000	1994
Buick	Century	22066	Virginia	$3,000	30,000	1995
Cadillac	Escalade	33156	Florida	$5,000	40,000	1996
Chevrolet	S-10	90001	California	$7,000	50,000	1997
Ford	Explorer	58259	North Dakota	$8,000	60,000	1998
Juguar	XJ	98112	Washington	$10,000	70,000	1999
Honda	Accord	60611	Illinois	$12,000	80,000	2000
Lincoln	LS	63463	Missouri	$15,000	90,000	2001
Volvo	V70	31626	Georgia	$20,000	100,000	2002

(b) Used Car Domain.