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

Jiying Wang*
Computer Science Department
Hong Kong Univ. of Science and Technology
Hong Kong
cswangjy@cs.ust.hk

Ji-Rong Wen
Information Management & System Group
Microsoft Research Asia
Beijing, China
jrwen@microsoft.com

Fred Lochovsky
Computer Science Department
Hong Kong Univ. of Science and Technology
Hong Kong
fred@cs.ust.hk

Wei-Ying Ma
Information Management & System Group
Microsoft Research Asia
Beijing, China
wyma@microsoft.com

Abstract

In a Web database that dynamically provides information in response to user queries, two distinct schemas, interface schema (the schema users can query) and result schema (the schema users can browse), are presented to users. Each partially reflects the actual schema of the Web database. Most previous work only studied the problem of schema matching across query interfaces of Web databases. In this paper, we propose a novel schema model that distinguishes the interface and the result schema of a Web database in a specific domain. In this model, we address two significant Web database schema-matching problems: intra-site and inter-site. 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 a unified solution to the two problems based on query probing and instance-based schema matching techniques. Using the model, a cross validation technique is also proposed to improve the accuracy of the schema matching. Our experiments on real Web databases demonstrate that the two problems can be solved simultaneously with high precision and recall.

1. Introduction

Besides web pages crawlable by specific URLs, the Web also contains a vast amount of non-crawlable content. This *hidden* part of the Web is comprised of a large number of online *Web databases* consisting of a searchable interface (usually an HTML form) and a backend database, which dynamically provides information in response to user queries [5] [13]. As compared to the static surface Web, the hidden Web contains a much larger amount of high-quality (often structured) information [8].

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

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

* This work was carried out when the author was visiting at Microsoft Research Asia.

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

schemas of Web databases and matching semantically-related attributes between them.

Previous approaches [16], [17], [21] to Web database schema matching primarily focus on matching query interfaces (i.e., on inter-site interface schema 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 Amazon.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 about Web databases. First, improper[2] queries often cause search failure or no returned results. Second, the keywords of proper queries that return results 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 keywords in the returned result data, and finally identifies the semantically corresponding attributes from both the interface and result schemas based on the previous analysis.

Using a domain-specific global schema, we present a combined schema model that can describe five kinds of schema matching for Web databases in the same domain: global-interface, global-result, interface-result, interface-interface, and result-result. The 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. Using the model, we also present a cross validation technique that improves the accuracy of the schema matching results.

The main contributions of this paper are:

- Introduction of a novel schema model of a single Web database that distinguishes what information users can query and what information users can browse.
- 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).
- Benefiting from the above generative view, introduction of a cross validation technique based on an approximate solution of the 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 our model with five schema matchings 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 approach. Finally, we give our conclusions and future work in section 7.

2. Combined Schema Model

A Web database is usually comprised of a query interface and a backend database. When a user query is submitted through the query interface, the site accesses its backend database for relevant data and returns the results to the user. Specifically, the query interface of the Web database usually contains multiple input elements, each of which may be associated with a schema attribute 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 Web database interface and the returned results partially reflect the schema of the backend database, but in different ways.

For instance, Figure 1 shows an example of an online bookstore[4]. The part labelled *Data Attributes* shows a possible schema of the backend database consisting of six[5] attributes {Title, Author, Publisher, ISBN, Format, Publication Date}. The part labelled *Interface* shows the query interface, which contains five input elements with surrounding text describing their semantics. When the keyword query "Harry Potter" is submitted through the "Title" element in the interface, a result page is returned by the web site containing its answer to the query (labelled *Result* in Figure 1 and containing three book instances with associated attribute values).

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 may provide an element in the interface for users to search on a particular data attribute (e.g., "ISBN"

[2] "Proper" means that the semantics of the query keywords 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.

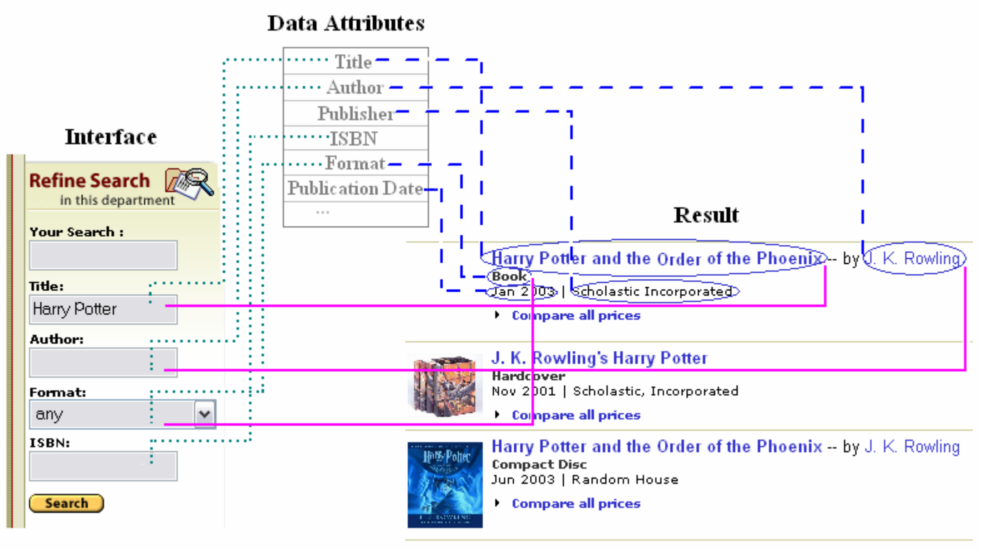


Figure 1. An example of a Web database with its search interface and result page.

element), this attribute's data values may not appear in the result pages. Likewise, the returned results may have attributes that users cannot query in the interface (e.g., Publisher attribute). Furthermore, Figure 1 shows three kinds of semantic correspondence represented by different line styles (dotted, dashed and solid). They are respectively, the correspondence between data attributes of the primary schema and elements in the query interface , the correspondence between the data attributes of the primary schema and instance values in the result pages, and the correspondence between elements in the query interface and instance values in the result pages.

In the deep Web, the primary schema of a Web database is hard to obtain directly as it is hidden behind query interfaces. However, previous work [16] makes the significant observation that, by examining the query interfaces of Web databases, an underlying generative global schema can be discovered for related Web databases in a specific domain. 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 Web database and propose a combined 3-layer schema model for matching the 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., Book or Used-car) or on the same topic (e.g., Job). 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. 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 (between RS and RS). Such *inter-site schema matching* is illustrated in Figure 2 as dashed edges between homogenous nodes of different sites.

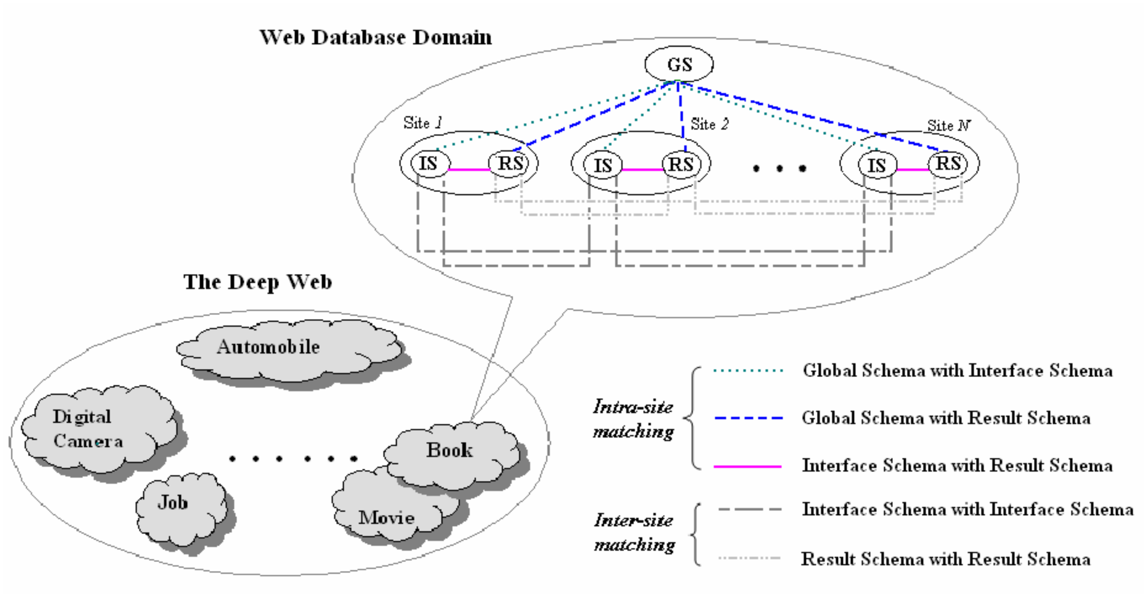

Web Database Domain
GS
Site 1
Site 2
Site N
IS
RS
The Deep Web
Automobile
Digital Camera
Book
Job
Movie
Intra-site matching
Global Schema with Interface Schema
Global Schema with Result Schema
Interface Schema with Result Schema
Inter-site matching
Interface Schema with Interface Schema
Result Schema with Result Schema

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

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

- **Automatically understand the semantics of schema attributes.** If the attribute semantics of one particular schema are accurately identified or known beforehand, then the attribute semantics of other schemas can also be discovered as long as they are correctly matched to an identified one. Even if the semantics of one particular schema are somehow wrongly identified in a matching with another schema, there is still opportunity for correction when it is matched to other schemas.
- **Automatically extract relevant content from Web databases.** Crawling the massive information hidden behind the query 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 semantics (i.e., to automatically extract relevant Web database content).
- **Meta-search multiple Web databases.** In this model, related Web databases are categorized by their domain. With a meta-search interface built for each domain, users can simultaneously search multiple Web databases of the domain. 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 query interfaces. Finally, their query results are integrated and displayed to users according to the match among their result schemas.

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 on which they rely (see [22] for a survey). Label-based methods only consider the similarity between schema definitions or attribute labels of two databases. Instance-based methods, such as [12] and [18], depend on the content overlap or statistical properties, such as data range and pattern, to determine the similarity of two attributes.

Recent work ([16], [17], and [21]) on matching query interfaces of Web databases fall into the first category, based on identifying the descriptive text surrounding interface elements as the attribute labels 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 has seldom been employed in the deep Web scenario because of the difficulty of automatically acquiring database contents hidden behind query interfaces. Paradoxically, a key prerequisite for automatic data acquisition from the deep Web is to understand the semantics of query 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 keywords through query 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 Observations

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 through the "Author" element, while it reports 228 matches for the same query when submitted through the "Title" element. On the other hand, we observe that when a proper query that returns a result web page is submitted through 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" through 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 values of 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., to discover semantically associated elements in the 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 in 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 this 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 query interface, an element identification component first locates qualified input elements in the query 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.

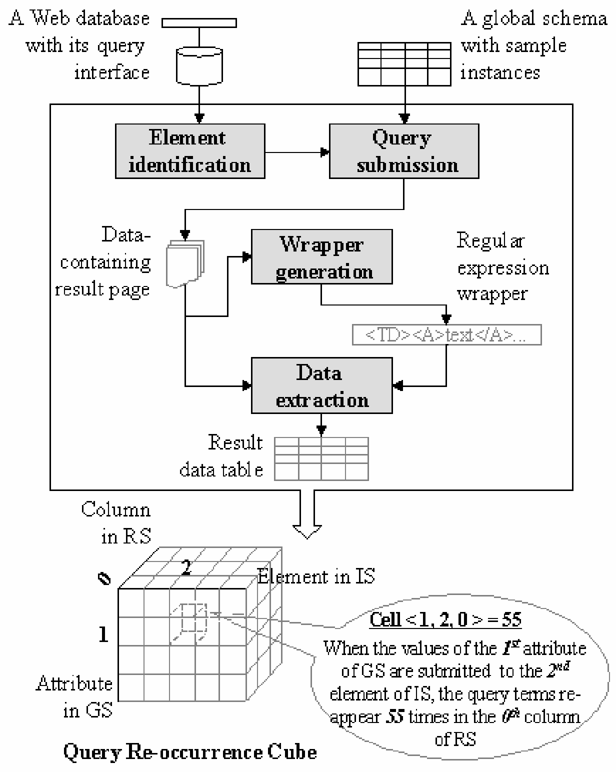

A Web database with its query interface
A global schema with sample instances
Element identification
Query submission
Data-containing result page
Wrapper generation
Regular expression wrapper
<TD><A>text</A>...
Data extraction
Result data table
Column in RS
0
2
Element in IS
1
Attribute in GS
Cell <1, 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 occurrence cube.

Given a Web database, the first task is to identify input elements in its query interface, which can be done by searching for the input-related tags[6] in a HTML searchable form. In the HTTP protocol, a query submission is carried out by sending a query request to the server containing the names of input elements and their corresponding query keywords. In this paper, we only submit one value to one element each time while keeping the default values for the other elements. We will consider the complex submission of querying multiple elements at one time in future work.

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 elements (i.e., we can only choose one or more of its OPTION values as the query keywords). 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.

After sending queries to the identified interface 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 that has attracted increasing research interest. Recently, attempts have been 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], [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 (i.e., of the result schema).

3.2.2 Query Occurrence Cube

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 given global schema. The cube width represents the number of interface elements considered (i.e., attributes of the interface schema). The cube depth is the number of columns in the result table (i.e., attributes of the 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 all given values for the *1st* attribute of GS are submitted to the *2nd* element of IS, the query keywords 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 and GS, IS and RS, and GS and 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 projection 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, $OM^{IR}_{M\times L}$ for IS and RS, and $OM^{GR}_{N\times L}$ for GS 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, IS and RS, and GS and RS based on the obtained matrices: $OM^{IG}_{M\times N}$, $OM^{IR}_{M\times L}$, and $OM^{GR}_{N\times L}$.

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$_{IS}$}.

EXAMPLE 1:

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

In fact, there are some properties of the occurrence matrix to consider when searching for the correspondence

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

[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.

or correlation between its rows and columns that represent 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) has 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 value (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 matrix element for Publisher$_{IS}$ and Publisher$_{GS}$ (468) does not have the highest value among the elements for Publisher$_{GS}$. However, it is relatively larger than the values of other matrix elements in the row for Publisher$_{IS}$.

Interestingly, we can view the schema-matching problem as follows. By sending sample queries, a part of the database content relevant to the queries is fetched from the Web database. For any two schemas, S_1 and S_2, of one 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, … A_n and the partitions by the attributes of S_2 are B_1, B_2, … 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 values re-appearing in the two partitions. The schema-matching problem now becomes that of finding the pair of partitions that belong to two schemas (e.g., A_i and B_j) such that their overlap with each other is more than their overlap with other partitions belonging to the opposite schema (e.g., *A_i and B_k or A_k and B_j*).

To help solve this problem, we employ the concept of *mutual information*, which 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 Y 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 X and Y 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 $\sum_j m_{ij}$ and m_{+j} being $\sum_i m_{ij}$. Note that if m_{ij} equals to 0, EMI is assumed to be 0 as well.

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:

$$\begin{array}{r c} & \begin{array}{cccc} \text{Title}_{GS} & \text{Author}_{GS} & \text{Publisher}_{GS} & \text{ISBN}_{GS} \end{array} \\ \begin{array}{r} \text{Author}_{IS} \\ \text{Title}_{IS} \\ \text{Publisher}_{IS} \\ \text{Keyword}_{IS} \\ \text{ISBN}_{IS} \end{array} & \begin{bmatrix} -0.007 & \boxed{0.019} & 0.010 & 0 \\ \boxed{0.033} & -0.005 & -0.001 & 0 \\ -0.004 & -0.003 & \boxed{0.025} & -0.001 \\ -0.002 & 0.001 & -0.011 & 0.029 \\ 0 & 0 & 0 & \boxed{0.055} \end{bmatrix}_{5\times 4} \end{array}$$

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 related attributes will have a larger overlap between each other than their overlap with other attributes of the other schema, as shown by the gray rectangles. For example, *EMI*(Author$_{IS}$, Author$_{GS}$) = 0.019 is the largest value 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 $e_{ij}\ge e_{ik}\mid k\ne j$ and $e_{ij}\ge e_{kj}\mid k\ne i$.

4.2 Inter-site Schema Matching

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

Borrowing the idea of *vector similarity* used in the *Vector Space Model* of Information Retrieval [2], we propose an approach to match interface/result schemas of different Web databases by computing their vector similarity. In the vector space model, documents are represented as vectors in a multi-dimensional space. In this 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.

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. Therefore, we can calculate the similarity (i.e., semantic correspondence) between attributes from different schemas by measuring their vector similarity. The following definition quantifies the inter-site schema matching between two Web databases.

DEFINITION 4: Given two query occurrence matrices of two Web databases' interface/result schemas $OM^{S_1 G} = [a_{ij}]_{n\times m}$ and $OM^{S_2 G} = [b_{ij}]_{l\times m}$ 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

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

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 whose value is the largest both in its row and column represents a match. For instance, Example 3 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 = {$Title_2$, $Author_2$, $ISBN_2$}. The grey rectangles depict the largest similarity values among rows and columns, which is also the correct matching. Interestingly, although the second attribute of IS_2, $Author_2$, is wrongly matched to $Publisher_2$ of GS in the previous intra-site schema matching (underlined element in EMI matrix of S_2), our method still can find the right inter-site matching.

EXAMPLE 3:

Occurrence Matrix of S_1

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

Occurrence Matrix Of S_2

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

***EMI* Matrix Of S_2**

$$\begin{bmatrix} 0.045 & -0.003 & -0.020 & 0 \\ -0.016 & 0.006 & \underline{0.032} & 0 \\ 0 & 0 & 0 & 18 \end{bmatrix}_{3\times 4}$$

Vector Similarity Matrix of S_1 and S_2

$$\begin{array}{c} \\ A_1 \\ T_1 \\ P_1 \\ K_1 \\ I_1 \end{array} \begin{array}{c} \begin{array}{ccc} T_2 & A_2 & I_2 \end{array} \\ \begin{bmatrix} 0.839 & \boxed{0.996} & 0 \\ \boxed{0.955} & 0.843 & 0 \\ 0.717 & 0.952 & 0.004 \\ 0.721 & 0.665 & 0.663 \\ 0 & 0 & \boxed{1.000} \end{bmatrix}_{5\times 3} \end{array}$$

4.3 Cross Validation

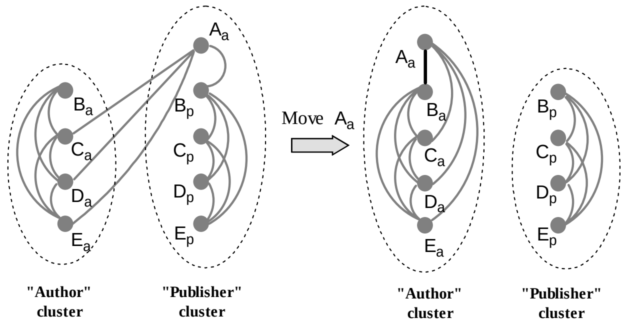
Given multiple Web databases in the same domain, we can employ the techniques proposed in sections 4.1 and 4.2 to identify the matching attributes belonging to schemas of an individual Web database and the matching attributes belonging to schemas of different Web databases. Consequently, we can employ the five types 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 target Web databases, we can categorize the IS (or RS) attributes into multiple clusters with respect to the GS attributes to which they have been matched. For example, the attributes, which are previously matched to the attribute A_G of the global schema, are categorized into one cluster, while the attributes, which are previously matched to the attribute P_G of the global schema, are categorized into another cluster. Recall that attributes are also matched to each other in inter-site schema matching. In the ideal case, an attribute in one cluster only matches with attributes in the same cluster. When a matching 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 matching is 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 [14]. 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 decreases. Accordingly, 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 first discarding it and then rematching 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 on the left depict how the attributes are primarily clustered according to which GS attribute they are matched (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 Used-car = {Make, Model, Postal-zip, State, Price, Mileage, Year}. We also manually collected 20 book instances and 10 car instances (details can be found in [25]) 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 our 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 Elements	#TS	%SS	#Result Columns	#Extracted Data
Book	4.2	343.3	32%	6.25	1322.9
Car	6.0	123.1	72%	5	995.3

The columns #TS and %SS of Table 1 represent, respectively, the number of total submissions made to the test Web databases and the corresponding success rate[9]. The reason that the Used-car domain has a lower number of submissions and a higher success rate than the Book domain is because SELECT and TEXTBOX input elements were treated differently when submitting the queries. We exhaustively tried all the attributes of the pre-known instances for a TEXTBOX element, while we only submitted the OPTION values of a SELECT element if they were 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, this domain has a higher number of submissions, but a lower success rate.

5.2 Matching Results

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) 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 used as the metrics. Precision 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.

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

Recall 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. 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 no "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 Used-car 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-appearance 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 keyword, such as "$10,000".

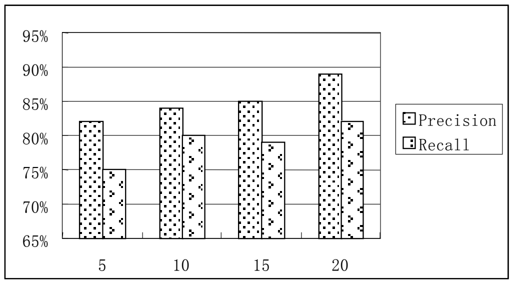


Figure 4. Result achieved by different number of sample instances.

We show in Figure 4 how the achieved results vary when the number of sample instances is increased. The 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 as the number of sample instances increases. However, more sample instances mean more query submissions to the Web database. 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 submissions and the achieved results.

Table 3. Inter-site schema matching results.

		IS-IS		RS-RS	
		P	R	P	R
Book	Label-based	90%	87%	95%	14%
	EVS	91%	71%	94%	86%
Car	Label-based	89%	88%	98%	25%
	EVS	92%	72%	89%	66%

In Table 3 we compare the inter-site schema matching results achieved by our proposed approach (EVS) based on *vector similarity* analysis to the matching results achieved by label-based approaches. Label-based approaches are mainly based on finding the synonym relationship between attribute labels. In matching interface schemas, we manually identified the surrounding text of input elements as their labels [16], [17], [21]. In matching result schemas, we manually found either explicit author-supplied column headers in the result pages or the text strings commonly shared by all extracted instances as the 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 since attribute labels are often unavailable in result pages. In addition, our approach does not require intelligent layout analysis to precisely identify the correct attribute labels.

Table 4. Effectiveness of cross validation.

		Before CV		After CV	
		P	R	P	R
Book	IS – GS	80%	70%	96%	83%
	RS – GS	95%	88%	98%	91%
	IS – RS	93%	87%	97%	90%
	IS – IS	91%	71%	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%

We present in Table 4 the effectiveness of the proposed cross validation approach in improving the overall accuracy. 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 as high a recall as we can precision (over 90%). We believe that the cause is not due to the ineffectiveness of the cross validation, but is due to the nature of the probing-based approach itself.

5.3 Discussion

In our experiments, we observed some issues that need further consideration.

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 they cover and their attribute-distinguishing capability. 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 result data 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-distinguishing 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 some 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 [11] and [22] for surveys). Existing work that addresses the problem of automatic schema matching for Web databases adopts the prior techniques on matching schemas of traditional databases. [16] presented a statistical approach to integrate the interface schemas of Web databases in the same domain. It hypothesizes that given Web databases 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 are label-based.

[17] introduced a tool, WISE-Integrator, that performs automatic integration of Web search interfaces in a product domain. WISE-Integrator employs comprehensive meta-data, such as element labels and default value of the elements, to automatically identify matching attributes from different search interfaces.

[19] 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.

[18] 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.

[11] 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. 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 work 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. Moreover, the instance-based schema-matching method is seldom used for schema matching in the Web database context since it is hard to get instances from Web databases. Supplied with a set of sample instances, our work proves that instance-based methods can also be 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 the various schemas associated with a Web database 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. We then investigate a unified solution to the two problems based on domain-specific query probing and attribute content overlap. Our instance-based approaches, which adopt the mutual information concept and vector similarity analysis, are quite powerful for precisely identifying the matching relationships among attributes of Web databases' interface and result schemas. Benefiting from our general framework, a cross validation technique, converted to a graph-partitioning problem, is introduced and shown to improve the matching performance.

Currently 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 work with previous label-based approaches to build a more robust matching system. In addition, we plan to extend this work to handle not only 1:1 mappings but also 1:N mappings over Web database schema attributes.

Acknowledgements

This work was partially supported by the UGC Research Grants Council of Hong Kong under the Areas of Excellence—Information Technology program.

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

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

[13] 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.

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

[15] F. Hakimpour, and A. Geppert. *Global schema generation using 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] W. Li and C. Clifton. *Semantic integration in heterogeneous databases using neural networks.* Proc 20th VLDB Conf, 1994.

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

[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.

[25] http://www.cs.ust.hk/~cswangjy/vldb04_exp.htm/