Available online at www.sciencedirect.com

ScienceDirect

Data & Knowledge Engineering 61 (2007) 281–303

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Combining schema and instance information for integrating heterogeneous data sources

Huimin Zhao [a,*], Sudha Ram [b]

[a] Sheldon B. Lubar School of Business, University of Wisconsin-Milwaukee, P.O. Box 742, Milwaukee, WI 53201, USA
[b] Department of Management Information Systems, Eller College of Management, University of Arizona, Tucson, AZ, USA

Received 26 March 2006; accepted 10 June 2006
Available online 10 July 2006

Abstract

Determining the correspondences among heterogeneous data sources, which is critical to integration of the data sources, is a complex and resource-consuming task that demands automated support. We propose an iterative procedure for detecting both schema-level and instance-level correspondences from heterogeneous data sources. Cluster analysis techniques are used first to identify similar schema elements (i.e., relations and attributes). Based on the identified schema-level correspondences, classification techniques are used to identify matching tuples. Statistical analysis techniques are then applied to a preliminary integrated data set to evaluate the relationships among schema elements more accurately. Improvement in schema-level correspondences triggers another iteration of an iterative procedure. We have performed empirical evaluation using real-world heterogeneous data sources and report in this paper some promising results (i.e., incremental improvement in identified correspondences) that demonstrate the utility of the proposed iterative procedure.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Heterogeneous databases; Data integration; Semantic correspondence

1. Introduction

The integration of heterogeneous data sources is becoming critical in modern organizations. Many organizations have developed a variety of heterogeneous data sources over time for different operational purposes and need to integrate these data sources for strategic purposes. Business mergers and acquisitions have further increased the appearance of heterogeneous data environments and, therefore, the need for data integration. In addition, many cooperating enterprises and business partners need to share and exchange information across their information systems. Some cooperation among interorganizational information systems may even be nationwide. For example, the National Institute of Standards of the United States has funded the development of a massively distributed Master Patient Index (MPI) [3], which is intended to allow all care providers access to the medical records of all patients in the US.

* Corresponding author. Tel.: +1 414 229 6524; fax: +1 414 229 5999.
E-mail address: hzhao@uwm.edu (H. Zhao).

0169-023X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2006.06.004

The need for semantic interoperability across heterogeneous data sources has also been amplified by the rapid proliferation of the Internet. The numerous data sources on the World Wide Web pose new challenges and opportunities for data integration. For example, the electronic product catalogs (E-catalogs) of different vendors need to be integrated [31]. Motivated by the promises of electronic commerce, many vendors have set up online storefronts, where customers can browse their product catalogs and buy product items. Buyers now can easily search and browse product information provided by many vendors in order to reach better purchasing decisions. Since the product catalogs of different vendors are semantically heterogeneous, however, buyers must still manually integrate and evaluate the product information obtained from different vendors.

To integrate a collection of heterogeneous data sources, either virtually or physically, the first and arguably the most critical step is to identify the *semantic correspondences* between every pair of the data sources, on both the *schema* level (for the purpose of *schema integration*) and the*instance* level (for the purpose of *data integration*). The question we need to answer before we can integrate the data sources is: What are the common concepts and overlapping data among the data sources?

Detecting semantic correspondences from heterogeneous data sources in large data integration projects is typically very time consuming and requires extensive human interaction. For example, Clifton et al. [8] reported on a project performed by the MITRE Corporation over a period of several years to integrate information systems that had been developed semi-independently over decades for the US Air Force. They found that tremendous time and effort were required just to determine attribute correspondences. Each of the databases contained from several hundred to several thousand attributes. Two matching tools, named DELTA and SemInt, were used to identify candidate attribute correspondences. DELTA computed a similarity measure between attributes based on their textual descriptions. SemInt clustered attributes into groups of similar ones based on several properties of the attributes, such as data types, length, and data patterns (summary statistics). The candidate attribute correspondences were then given to domain experts such as Air Force pilots and planners for review and confirmation. The time consumed by such manual review largely overshadowed the time spent in using the tools. It often took a few weeks for the investigator, local database administrators (DBA), and domain experts to communicate with each other in order to determine whether two attributes in different databases, e.g., mission start time and mission takeoff time, mean the same thing.

Detecting instance-level correspondences is often even more time consuming than detecting schema-level correspondences because there are typically more records than tables and attributes in real-world databases. The records are also updated more frequently than the schemas. For example, Winkler [47] reported the integration of several mailing lists for the US Census of Agriculture in 1992. An automated record-matching tool, based on Fellegi and Sunter's *record linkage theory* [15], was used to recommend potential matching records. These mailing lists were much simpler than the US Air Force databases described by Clifton et al. [8]. However, thousands of person-hours were still required for the clerks just to review and confirm those candidate matching records that the record matching tool was not able to determine with acceptable accuracy.

Much research has been conducted in the last two decades to facilitate the identification of semantic correspondences from heterogeneous data sources. Identification of schema-level correspondences has been referred to as *Interschema Relationship Identification* (IRI) [36]. Identification of instance-level correspondences has been given various names, including *instance identification* [40], *entity identification* [16], *merge/purge* [19], *approximate record matching* [45], and *record linkage* [15,47].

Most past research, however, has studied the two problems (i.e., identification of schema-level correspondences and identification of instance-level correspondences) separately, without considering the interaction between the two tasks. Research studying schema-level correspondences usually has started from scratch and generally has not taken instance-level correspondences into consideration. On the other hand, research studying instance-level correspondences usually has taken schema-level correspondences for granted and thus has not investigated how identified instance-level correspondences could be fed back to the schema level for improved understanding of schema-level correspondences.

It is our contention that combining analyses on the two levels in a synergistic way may improve the quality of identified semantic correspondences on both levels, because the two problems are inherently related. In this paper, we propose an *iterative procedure*, in which semantic correspondences on the two levels are identified alternately and incrementally. We also review promising techniques for each step of the procedure. We have evaluated our approach using real-world data and will present some empirical results to demonstrate the utility of our approach.

In this paper, we deal specifically with relational databases while assuming that techniques exist to transform, map, or wrap, either physically or virtually, other types of data sources (e.g., pre-relational databases, object-oriented databases, spreadsheets, text files, and semi-structured or unstructured data) into relational ones. We use the term "database" to refer to any such data source, and we use relational terminology (i.e., relation, attribute, and tuple).

The paper is organized as follows: in the next section, we briefly review some past approaches to detecting semantic correspondences. In Section 3, we describe our proposed iterative procedure and related techniques. We then report some empirical evaluation results in Section 4. Finally, we summarize the contributions of this work and discuss future research directions.

2. Related work

Past research has focused on either schema-level correspondences or instance-level correspondences. In this section, we briefly review some of the approaches and techniques proposed in the past.

Various approaches and techniques have been proposed for detecting schema-level correspondences across heterogeneous databases. First, some of them apply *linguistic techniques*, such as dictionary, taxonomy [5,41], thesaurus [30], conceptual graph, case grammar [2], and speech act theory [20], to measure the similarity between the names of schema elements. Such linguistic approaches require that the names of the schema elements be drawn from standard vocabularies that can accurately describe the meanings of the schema elements. These approaches are therefore difficult to apply in many legacy systems, where schema elements are not well named using standard terms. Second, some other approaches use *heuristic formulae* to measure the similarity between schema elements, based on the names and structures of the schema elements [18,27,29,32,38]. Such formulae are often derived from particular database integration applications and are therefore hard to generalize to other applications. Third, another approach computes the similarity between the text descriptions of schema elements (in design documents) using similarity measures developed in the *information retrieval* field [4]. The major difficulty with this approach is that the design documents are often out of date, inaccurate, or even not available in many legacy systems. Fourth, other approaches have used *cluster analysis* techniques to cluster schema elements based on the meta-data (e.g., name, schematic specification, summary statistics) about the schema elements [11,42,52,54]. However, due to various problems associated with the features for such cluster analysis [52,54], users must carefully evaluate the results generated by the cluster analysis techniques.

While the above-mentioned approaches use only meta-data, *statistical analysis* techniques such as *correlation* and *regression* analysis, which examine the actual instance data stored in the databases, have also been used to analyze the relationships among numeric attributes [14,25]. These statistical techniques are more rigorous than those used in other approaches that only examine meta-data. However, they require that some sample data from heterogeneous databases be integrated somehow, i.e., instance-level correspondences must be identified first.

Several approaches to detecting instance-level correspondences have also been proposed. Some approaches rely on users to provide decision rules for determining whether two tuples represent the same real-world entity [7,10,19,39]. Such rule-based approaches explicitly incorporate the domain knowledge of human experts. The major difficulty with these approaches is the so-called *knowledge acquisition bottleneck* in knowledge engineering. It is often very difficult for human experts to articulate their domain knowledge in the form of decision rules.

Classification techniques have been used to train classifiers for predicting whether two tuples from heterogeneous databases match or not. Fellegi and Sunter's record linkage theory [15], which extends the Bayes classification method, has been the foundation of many record linkage systems, including GRLS developed at Statistics Canada [13] and the US Bureau of the Census software for record linkage [48]. Other classification techniques, such as logistic regression [34] and decision trees [16,17,44,45], have also been used. The advantage of classification-based approaches over rule-based approaches is that decision rules are automatically learned from training examples rather than elicited from domain experts; it is relatively easier for human experts to provide some classified examples than to articulate the classification rules. However, one problem with previous classification-based approaches is that they all commit to a particular classification technique, which may not be the best in a particular situation.

Past research has studied the two problems, i.e., identification of schema-level correspondences and identification of instance-level correspondences, in isolation. In this paper, we propose an iterative procedure for detecting correspondences on the two levels alternately and incrementally.

3. An iterative procedure for detecting semantic correspondences

Identifying schema-level correspondences and identifying instance-level correspondences are two critical subproblems in database integration. Since the two problems are related, combining them into a comprehensive procedure may be beneficial. Classical statistical analysis techniques, such as correlation analysis and regression, have long been used to analyze the relationships among attributes of a single data set. If two relations can be integrated, we can use the same techniques to analyze the relationships among attributes from the two relations. Integrating two relations, however, requires us to understand instance-level correspondences. Comparing two tuples from different relations must be based on some common attributes. A better understanding of attribute correspondences between two relations enables us to compare instances across the relations more accurately; a better understanding of instance-level correspondences enables us to integrate the data and analyze attribute correspondences more accurately. It is evident that the two problems are inherently related. We can identify correspondences on the two levels in an iterative way and gradually improve our findings.

We then need to decide where to start: the schema level or the instance level. To compare two tuples from different relations, we must know what (i.e., common attributes) to compare. To compare attributes of two relations, however, does not require us to know common data or any data at all stored in the relations. Schemas are designed before data are populated. At least the designer of a database understands (and hopefully remembers) her intended meaning of each relation or attribute before the database is actually used. It is therefore possible to compare schemas somehow, although maybe rather expensively and inaccurately, before we compare actual data.

We can now envision a procedure in which we analyze correspondences on the two levels (i.e., schema level and instance level) in an iterative way. Fig. 1 shows our proposed iterative procedure for detecting semantic correspondences on both the schema level and the instance level. The procedure consists of the following major steps: (1) an initial set of schema-level correspondences (corresponding tables and corresponding attributes) across the heterogeneous databases under investigation are identified. (2) Instance-level correspondences (corresponding records) are then identified by comparing the corresponding attributes of the records

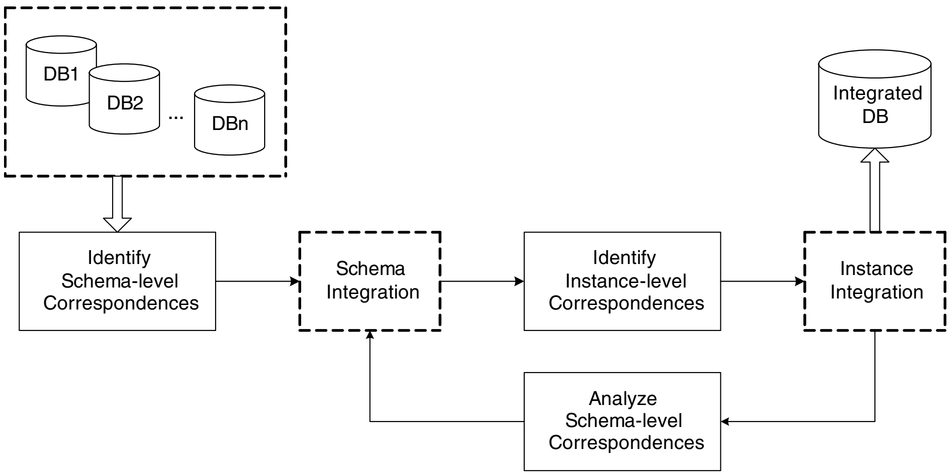

DB1
DB2
...
DBn
Integrated DB
Identify Schema-level Correspondences
Schema Integration
Identify Instance-level Correspondences
Instance Integration
Analyze Schema-level Correspondences

Fig. 1. An iterative procedure for detecting semantic correspondences.

across the corresponding tables. (3) As instance-level correspondences have been identified and data combined into a single data set, schema-level correspondences can be more accurately evaluated using statistical analysis techniques such as correlation and regression. Previously identified correspondences are verified. Other possible combinations of attributes are then explored to detect missed potential correspondences. Significant improvement in the identified schema-level correspondences triggers another iteration of the procedure.

Such an iterative procedure reduces the requirements for the accuracy of individual steps. Although each step does not produce a perfect result, the iterative procedure allows the result to be improved gradually. The user can decide to terminate the procedure when no significant improvement in the identified semantic correspondences is found during any iteration. Note that the above-mentioned steps for identifying schema-level and instance-level semantic correspondences can be streamlined with schema integration [36] and instance integration steps to build the ultimate integrated database with an integrated schema. However, in this paper, we have chosen to focus on the steps for identifying semantic correspondences.

Now we need to choose appropriate techniques for the initial identification of schema-level correspondences and the identification of instance-level correspondences. As we hope to automate the process as much as possible, we prefer to use learning-based techniques. To decide whether *unsupervised* (i.e., *clustering*) or *supervised* (i.e., *classification*) learning techniques are suitable for each of the two levels, we need to compare the characteristics of the two levels. We assume for most databases that: (1) the number of schema elements (i.e., relations and attributes) is relatively smaller than the number of tuples and (2) schemas are relatively more stable than instances. The size of the schema of a database is usually much smaller than the size of its data. If there are hundreds of relations and thousands of attributes in a database (e.g., in the US Air Force databases [8]), there may be millions of tuples. Schemas do change over time, but they evolve much more slowly than instances.

We feel that cluster analysis techniques are more suitable for the identification of schema-level correspondences and classification techniques are more suitable for the identification of instance-level correspondences. Cluster analysis techniques group similar examples into rough groups and do not support forecast on unseen examples. They need to be rerun in response to changes in the data to be analyzed. Classification techniques produce very specific (match or non-match) results and can predict the classes of unseen examples. As supervised learning, classification requires the user to provide previously classified examples.

Cluster analysis is more suitable than classification for the identification of schema-level correspondences because: (1) some amount of follow-up manual evaluation is affordable given the relatively small sizes of the schemas. Cluster analysis results must be carefully evaluated in light of domain knowledge and semantics that reside only in the user's mind. (2) Schemas are relatively stable compared to instances. There is no requirement for prediction on unseen data unless the schemas change or a new schema needs to be analyzed. (3) The benefits of prediction may not justify the costs of providing training examples and training classifiers, due to the small sizes and the relative stability of the schemas. (4) Results produced by any technique cannot be precise, given the limited available semantic features of schema elements. If classification techniques are used instead, error rates may be too high.

Classification is more suitable than cluster analysis for the identification of instance-level correspondences because: (1) the amount of manual evaluation must be reduced as much as possible, given the relatively large number of tuples. For the technique to be practically useful, the results produced by the technique must be more specific than some rough groups, as generated by clustering techniques. (2) Data are changing frequently. It is not wise to rerun clustering techniques whenever the data are updated. In some situations, updated data even need to be analyzed in real time. (3) Considering the large number of frequently updated tuples, it is worthwhile to prepare training examples to train classifiers, so that the rest of the tuples not included in the training example and the new data in the future can be automatically classified.

We are not suggesting that cluster analysis does not apply to identifying instance-level correspondences and classification does not apply to identifying schema-level correspondences. However, for most real world situations, we feel that cluster analysis is more appropriate than classification for identifying schema-level correspondences and classification is more appropriate than cluster analysis for identifying instance-level correspondences.

The proposed procedure is not intended to be totally automated. Automated techniques recommend potential correspondences to the user to reduce the amount of human interaction but may not automatically make decisions on the user's behalf. The user may be responsible for reviewing and confirming (or rejecting) the recommended potential correspondences, utilizing human domain knowledge to do so. The accuracy of the final merged database depends on the quality of the user's scrutiny. Human intervention may be involved in every step of the procedure. Cluster analysis is highly empirical in nature and requires careful evaluation and interpretation of the results. Any classification rules are faced with a tradeoff between the two types of errors, false matches and false non-matches. In practice, to reduce the two types of errors, we may use classification techniques to designate some of the matches and non-matches and leave some hard examples for users to review manually. Highly correlated attributes detected by statistical analysis techniques may describe different properties about some entity type. Potential semantic correspondences suggested by such statistical analysis techniques need to be verified by users in light of domain knowledge. Our objective is to reduce manual evaluation as much as possible rather than to totally eliminate it. In addition, we do not preclude any manual effort, e.g., manual comparison and integration of the ER diagrams of the databases. Such manual analysis may be sufficient in relatively simple applications. The termination of the procedure is also the user's call; the user can decide to terminate the procedure when no significant improvement in the identified semantic correspondences is observed during any iteration.

In the following subsections, we will discuss the techniques that can be applied in the different steps of this iterative procedure. Note that the techniques for clustering similar schema elements and those for classifying tuple pairs (into match and non-match) have been described previously [52–54] and are briefly reviewed in this paper for the sake of completeness.

3.1. Cluster analysis techniques for identifying schema-level correspondences

In the initial identification of schema-level correspondences, we apply cluster analysis techniques to identify clusters of similar schema elements; they will then be evaluated by the users. Given a set of objects drawn from some problem domain, cluster analysis techniques group the objects into several groups (called clusters) of similar objects. Because the groups to be discovered are previously unknown, cluster analysis is called "unsupervised" learning.

In the input data set for clustering n schema elements (relations or attributes), each schema element is represented as a vector of m features. The data set is an $n \times m$ matrix,

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \cdots & \cdots & x_{nm} \end{bmatrix},$$

where x_{ij} is the value of the jth feature for the ith schema element. The similarity between two schema elements is measured using some measure of distance (e.g., Euclidean) between the vectors representing the schema elements. The objective of cluster analysis is to divide all the schema elements into several clusters such that schema elements within the same cluster are similar to each other while schema elements across clusters are dissimilar to each other.

The features may be weighted according to their relative importance in discriminating the schema elements. Potential features not included in the analysis are effectively given the weight zero. In practice, when it is difficult to assess precisely the relative importance of the features, equal weights are often applied to the features, following a normalization or standardization procedure. Normalization and standardization can, therefore, be viewed as a special form of weighting the features. A common practice is to standardize each feature to unit variance; every value of a feature is divided by the standard deviation of the feature. Another normalization method is to divide every value of a feature by the *range* of the feature. When there are many features, a dimensionality reduction technique, such as *principal component analysis* [1], can be used first to extract a smaller set of transformed features.

There are many statistical techniques for cluster analysis [12]. They are divided into two types: *hierarchical* and *nonhierarchical*. A nonhierarchical method, such as K-means, generates a predefined number of clusters.

A hierarchical method, *agglomerative* or *divisive*, simultaneously generates several clustering results on different granularity levels. An agglomerative method starts from the finest partition, in which each individual case is a cluster, and successively merges smaller clusters into bigger ones. A divisive method proceeds in the opposite direction; it starts from the coarsest partition, in which all the cases are in a single cluster, and successively separates the cases into finer clusters. For more details about these techniques, please see [12]. These statistical clustering techniques are widely available in statistical packages, including SAS, SPSS, SYSTAT, and BMDP.

There are also unsupervised neural networks, such as Kohonen's *Self-Organizing Map* (SOM) [22], for cluster analysis. We have developed an SOM tool for clustering schema elements [52,54] that displays the schema elements on a two-dimensional map using the U-matrix method [9]. Similar schema elements are located close to each other; the distances between neighboring elements are indicated by gray levels. The tool can also generate clustering results on any similarity level chosen by the user.

Past empirical studies (e.g., [12,28,33]) have not agreed on any universally best clustering method. Users are encouraged to apply multiple methods and compare the results for a particular application [1]. Users are also encouraged to carefully evaluate the clustering results in light of domain knowledge.

Information about schema elements that can be used as input features in clustering schema elements includes: similarity between schema element names, description in design documents, schema specification (e.g., data type, length, and constraints), data patterns (i.e., characteristics of values), and usage patterns [23,37]. However, there are problems associated with these potential input features. Phrases and acronyms rather than single words are more commonly used to name schema objects in real-world databases. The meaning of a schema object may change as the associated business process evolves. Design documents are often out of date, inaccurate, or even not available in real world practices. While names and documents directly describe the meanings of schema objects, the semantics are only indirectly reflected by schema specification, data patterns, and usage patterns. Due to such problems, users are encouraged to carefully evaluate and select appropriate features in each particular application.

Note that the two terms, attribute and feature, have often been used interchangeably in the literature. In this paper, we use the term "attribute" to refer strictly to an attribute in the heterogeneous databases under comparison and use the term "feature" to refer to a property (e.g., data type, length, data patterns, etc.) about the database attributes. In a data set for cluster analysis, the rows represent database attributes, while the columns represent features about the database attributes. The cluster analysis clusters the database attributes based on the features.

3.2. Classification techniques for identifying instance-level correspondences

On the instance level, it needs to be determined whether two tuples from different (or the same) relations correspond to the same real-world entity. This can be represented as a binary classification problem, where each pair of tuples needs to be classified as match or non-match based on a set of input features (independent variables). Each input feature is a similarity measure between the values of the two tuples for a semantically corresponding attribute pair. A decision model (called *classifier*) can be induced from a set of training examples (tuple pairs with known classes) and subsequently applied to predict the classes of other tuple pairs. Classification is characterized as *supervised* learning because the classifier is learned from a set of classified cases that have been provided by some human expert, known as the supervisor or teacher.

Many methods for learning classifiers have been developed in such fields as multivariate statistical analysis, machine learning, and artificial neural networks [46]. Some widely used statistical classification methods include naive Bayes, Fisher's normal linear discriminant analysis, logistic regression, classification via linear regression, and k-nearest neighbor [1,49]. Fellegi and Sunter's record linkage theory [15] is an extension of the Bayes classification method specifically for the record linkage problem. Machine learning classification methods usually learn decision tables, trees, or rules. C4.5 (an updated version, C5.0, is now available) is one of the most widely used decision tree techniques [35]. The simplest classification rules that use a single feature in the prediction are called 1-rule [49]. Artificial neural networks are sophisticated networks with an input layer (corresponding to the input features), an output layer (corresponding to the output class), and zero or more intermediate hidden layers. *Back propagation* is one of the most popular neural network techniques for classification problems.

Many of the above-mentioned classification techniques have been implemented in the Weka system [49]. The “no free lunch theorems” [50] have proven that there is no universally superior classification technique that performs best in every application. This conclusion has also been supported by the results of many empirical studies. For example, Weiss and Kulikowsli [46] empirically evaluated several widely used classification techniques. Lim et al. [24] empirically evaluated 33 classification algorithms, including twenty-two decision tree, nine statistical, and two neural network algorithms. Their results show that different methods may perform the best for different problems. In each classification task, multiple classification techniques can be experimented with to select the most appropriate one.

The input features used in classifying tuple pairs are usually similarity measures (called *attribute-matching functions*) for comparing semantically corresponding attributes. The simplest attribute-matching function is equality comparison, which returns 1 if two attribute values under comparison are equal and 0 otherwise. However, since there are often various data errors and data discrepancies across databases [26], two corresponding tuples may not match exactly on semantically corresponding attributes. More sophisticated approximate attribute-matching functions are then needed to measure the similarity between attribute values, while accounting for such errors and discrepancies. There are many methods (e.g., Levenshtein’s metrics) for measuring the similarity between strings [6,43]. Normalized distance functions can be defined to compare numeric attributes. Coding differences between corresponding categorical attributes across databases can be resolved using special matching dictionaries.

When multiple matching functions are available for comparing a pair of corresponding attributes, correlation analysis can be used to select the most predictive one for classification. The *correlation* between each attribute matching function and the class (match or non-match) of tuple pairs can be estimated using the training sample for classification. The attribute matching function that is most correlated with the class is then selected.

3.3. Statistical analysis techniques for evaluating schema-level correspondences

When some instance-level correspondences between two relations have been identified, data from the two relations can be integrated into a single relation. Statistical analysis techniques can then be applied to analyze schema-level correspondences. New findings about schema-level correspondences trigger a new iteration in the overall procedure for detecting both schema-level and instance-level correspondences.

We can use statistical analysis techniques, such as *correlation* and *regression*, to analyze the relationships among attributes originally from different databases. Simple correlation analysis and regression analysis techniques can be used to analyze the relationships between two attributes. The *Pearson product-moment correlation coefficient* is the most widely used index of correlation to measure the degree of the *linear relationship* between two interval or ratio variables. The Pearson correlation, denoted r, between two variables X and Y is defined as

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(Y_i - \overline{Y})^2}},$$

where $\overline{X}$ and $\overline{Y}$ are the means of the two variables, X_i and Y_i ($i = 1, 2, \ldots, n$) are the individual values of the two variables. A related statistic is the *coefficient of determination*, r^2, which is equal to the square of the correlation coefficient. r^2 measures the proportion of *variance* in one variable that can be explained by the linear relationship between the two variables.

For two highly correlated attributes, i.e., whose r^2 is above some selected threshold value, we use regression analysis to further analyze the relationship between the attributes. The simplest and most widely used regression analysis technique is *linear regression*. Linear regression finds a linear model to describe the relationship between two variables X and Y. The model can be expressed as $Y' = \alpha + \beta X$, where Y' stands for a predicted value for variable Y, and α and β are coefficients determined from the sample data. Note that the distinction between *dependent variable* and *independent variable* is arbitrary here; two attributes from different databases that are considered potentially related are *symmetric* in the regression model. For a pair of attributes under analysis, the regression analysis can be run twice, positioning each attribute on different sides of the model. Linear regression is useful in discovering measurement differences (e.g., the metric system vs. the US system for measures and weights) and scaling differences (e.g., number of dollars vs. number of thousands of dollars) across data sources.

If one attribute in a database is related to many attributes in another database simultaneously, *multiple correlation* and *regression* analysis should be used. The *multiple linear regression* model for a variable Y and a set of variables $X_1, X_2, \ldots, X_P$ can be expressed as $Y' = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_P X_P$. The *multiple correlation coefficient*, denoted R, is the correlation between Y and the predicted value Y', which is a linear combination of X_i's. The *coefficient of multiple determination*, R^2, indicates the proportion of variance in Y that can be explained by the combined predictors $X_1, X_2, \ldots, X_P$. The difference between R^2 of a multiple linear regression model and r^2 of a simple linear regression model indicates the improvement in predicting Y that can be achieved by using multiple X_i's instead of a single X. The correlation between Y and a particular X_i after adjusting for the linear effect of other variables is measured by a *partial correlation coefficient*.

If many attributes in a database are related to many attributes in another database simultaneously, *canonical correlation analysis* can be used. Given a set of variables $Y_1, Y_2, \ldots, Y_Q$ and another set of variables $X_1, X_2, \ldots, X_P$, canonical correlation finds a linear combination of Y_i's and a linear combination of X_i's, called *canonical variables*. The canonical variables can be expressed as

$$U = \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_Q X_Q \quad \text{and} \quad V = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_P X_P.$$

The coefficients α_i's and β_i's are chosen to maximize the correlation between the two canonical variables.

Pearson correlation coefficient and linear regression describe linear relationships among variables. There are also correlation measures (e.g., correlation ratio) and regression methods (e.g., quadratic regression) that describe nonlinear relationships among variables.

The techniques we have described so far are better suited to analyzing *interval* and *ratio* variables than *categorical*ones. There are correlation measures, such as the *Cramér's measure of association* and the *Spearman rank correlation coefficient*, for measuring the relationship between *nominal* or *ordinal* variables [21]. In general, however, it is harder to analyze relationships among categorical variables, due to various kinds of discrepancies among databases. Different (exact or approximate) attribute-matching functions, which we have described earlier, need to be tried to see whether two categorical attributes are related in some way. Given two attributes, if the *mean* of some attribute-matching measure is close to 1, the two attributes are likely to be related. We can further compute the correlation between an attribute-matching function to the match label (match or non-match) of tuple pairs to see the discriminating power of the attribute-matching function. A highly discriminating attribute-matching function shows more evidence that the two attributes are potentially related. The mutual information measure defined in the information theory can also be used to measure the statistical dependence between variables.

The techniques described above can be used in several ways. First, previously identified attribute correspondences can be verified using these techniques. This may reveal some false correspondences. Second, other possible combinations of attributes can be automatically explored to detect highly correlated and potentially corresponding attributes. Highly correlated attributes, however, may not semantically correspond with the same property of some real-world entity type. For example, the recommended retail price of a product and the actual price of the product offered by a particular retailer may be highly correlated but describe two different properties about products. Potential attribute correspondences suggested by statistical analyses need to be evaluated in light of domain knowledge. Third, hypothetical attribute correspondences in the analyst's mind can be tested using these techniques as well as hypothesis-testing techniques.

4. Empirical evaluation

We have evaluated our proposed approach to detecting both schema-level and instance-level correspondences using several sets of real-world data. In this paper, we report on some of our experiments using partial book catalogs that we have extracted from two leading online bookstores. One of the two bookstores (store A) listed the following sixteen attributes about books on its Web site: ISBN, author names, book title, series title, list price, our price, cover, type, edition, publishing month, publishing date, publishing year, publisher, number of pages, average rating, and sales rank. The other store (store B) listed the following 14 attributes: ISBN, book title, author list, retail price, our price, cover format, edition, number of pages, publisher, publishing month, publishing year, edition description, sales rank, and rating. Since the two stores did not display the names of some attributes on the Web sites, we named those attributes based on our intuition. The attribute names of catalog A were A.ISBN, A.Authors, A.Title, A.Series, A.List_price, A.Our_price, A.Cover, A.Type, A.Edition, A.Month, A.Date, A.Year, A.Publisher, A.Pages, A.Avg_rating, and A.Sales_rank. The attribute names of catalog B were B.ISBN, B.Title, B.Author, B.Retailprice, B.Ourprice, B.Coverformat, B.Edition, B.Pages, B.Publisher, B.Pubmonth, B.Pubyear, B.Editiondesc, B.Salesrank, and B.Rating.

4.1. Detecting schema-level correspondences

We used two statistical clustering methods, including K-means and hierarchical clustering, as well as the SOM tool we had developed, to cluster the attributes of the two book catalogs, based on a set of features. We selected some features for calculating the distances between the attributes. Because the sample data were extracted from the Web sites without direct access to the backend databases, we could not use any design documents, schematic specifications, or usage patterns. We could only compare the attribute names and data patterns (summary statistics). We intuitively assigned a similarity degree (in the range of $[0, 1]$) for each pair of attribute names, and we estimated some data patterns about each attribute using two sample data sets extracted from the Web sites. The 14 data patterns consisted of some summary statistics (mean, standard deviation, max, and min) about the lengths of values, some summary statistics about the percentages of digits in the values, some summary statistics about the percentages of letters in the values, the percentage of values that were missing, and the ratio of the number of distinct values to the total number of tuples.

Note that although not applicable in this example, other features, including data types, length, primary key, foreign key, value and range constraints, and access restrictions, have been suggested in previous research [8]. Some of these features may not be numerical-valued. However, dummy variables (e.g., converting a Boolean value into 1 or 0) can be generated for such features and incorporated into the cluster analysis.

We linearly normalized the features into the range of $[0, 1]$ and then performed Principal Component Analysis on the features to obtain a set of principal components with a reduced dimensionality. While the number of data patterns is not affected by the number of attributes, the number of similarity degrees between attribute names is proportional to the number of attributes. The dimensionality of the features needs to be reduced when there are many attributes in the databases. Principal Component Analysis is a statistical technique frequently used for dimensionality reduction [1]. It derives a sequence of orthogonal linear combinations of the original features, referred to as principal components. These principal components are eigenvectors of the covariance matrix of the original features. They are ordered (in descending order) by the amount of variance in the original data that they account for. The first few principal components that account for most of the variance in the data are retained, while the others are discarded.

There were 14 data patterns and 30 similarity degrees between attribute names in this case. The Principal Component Analysis procedure in SPSS with the default selectivity setting (eigenvalues > 1) extracted 15 principal components from the 44 original features. The input data set for clustering attributes consisted of 30 rows and 15 columns. Each row of the data set represented one of the 30 attributes in the two catalogs. Each column represented a principal component, which was a linear combination of the 44 original features describing the attributes of the catalogs.

The detailed clustering results are available in [52,54]. Since a hierarchical representation helps users to incrementally evaluate the results, we ran the nonhierarchical method K-means several times (with different Ks) to simulate a hierarchical clustering result. However, there were some conflicts in the results on different Ks; when $k = 15$, A.Edition, B.Edition, A.Month, and B.Pubmonth were grouped into the same cluster, but when $k = 10$, they were grouped into different clusters.

While we did not find significant difference among the three methods in terms of accuracy, SOM intuitively appeared better than the other two for visualizing the results. Similar attributes were located close to each other on the SOM map; different clusters were separated by dark boundaries. Clustering results on different similarity levels were immediately shown when different similarity thresholds were specified using a convenient slider.

In this case, we subjectively assigned similarity scores among attribute names. All other features can be obtained automatically. We also ran the same clustering methods without using the assigned similarity scores and found that the quality of the clustering results was reduced; the boundaries between clusters became vaguer. This shows that attribute names (and descriptions) that directly describe the meanings of attributes are important in comparing attributes. We recommend that users incorporate attribute names and descriptions whenever they are available and comparable. Unfortunately, attribute names are not always easily comparable, due to the various problems (e.g., synonym/homonym, abbreviations, meaning changes, etc.) described previously, and descriptions may not be available in real-world situations. Users are not required to manually provide similarity scores. However, if only data patterns are available for the cluster analysis, we remind users to exercise more caution in reviewing the results. In addition, string similarity measures [6,43] can also be tried to measure the similarities among attribute names. However, we remind users that such measures may not adequately reflect semantic similarity.

In this case, there was only one relation in each catalog. When there are many relations in each database, the same clustering techniques can be used to cluster relations, too. Potential features for clustering relations include similarity between names or descriptions of relations, cardinality (number of attributes), degree (number of tuples), relationships (number of foreign keys and number of referencing relations), and usage patterns (e.g., update frequency and number of users or user groups) [42]. The patterns of the attributes of a relation can also be further summarized to generate patterns of the relation. It is possible to start with either clustering relations or clustering attributes. When corresponding relations have been found, and detecting corresponding attributes can be restricted to only corresponding relations, the search space is reduced. On the other hand, when attributes have been clustered, the clustering results can be used to help identify corresponding relations, too. We do not presume a particular order for these two steps. However, note that there are usually many fewer relations than attributes. Comparing relations tends to be less time-consuming than comparing attributes. In the Air Force example [8], while there are between several hundred and several thousand attributes, there are only between one hundred and three hundred relations in each database. Consequently, detecting relation correspondences was not reported in [8]. Comparing relations first, therefore, is likely to be a more economical choice.

Generally speaking, errors, both false positives and false negatives, are unavoidable. This is the reason for adopting an iterative procedure to detect correspondences on the two levels alternately. We describe how we used statistical analysis techniques to reevaluate attribute correspondences in Section 4.3.

4.2. Detecting instance-level correspondences

On the instance level, we wanted to decide whether a pair of tuples (i.e., records) in the two catalogs referred to the same book. We trained classifiers to determine whether or not a pair of tuples matched, based on their distances on the common attributes identified in the previous cluster analysis. There was a common *key* attribute, i.e., ISBN. Records about the same book had the same ISBN value even if they were stored in different databases. Therefore, the identification of corresponding records from these two catalogs is simple; we can rely on the ISBN to match the records. We chose this case for empirical evaluation because the ISBN could provide convenient and objective training and testing data to evaluate various classification techniques. We trained classifiers based on other attributes common to the two catalogs while withholding the ISBN, which served as the determinant of the correctness of the results. In the integration of general product catalogs, however, when there is no common key, domain experts must manually match some records to train and test the classifiers. There are other interesting situations where some records have a key value that is common across catalogs while other records do not. In such cases, the records that have a common key value can be used to train classifiers, which are then used to compare other records.

The comparison of a pair of tuples was based on thirteen pairs of corresponding attributes, such as title (A.Title and B.Title) and list price (A.List_price and B.Retail_price), which had been identified in the previous cluster analysis and confirmed in the follow-up evaluation. We randomly generated a balanced data set with 1404 match tuple pairs and 1404 non-match tuple pairs to train and test classifiers.

After experimenting with the training data set, we defined various attribute-matching functions. Table 1 summarizes these functions. All functions returned a value between 0 and 1. We assigned the minimum similarity degree 0 to a pair of attribute values if one or both of them were *missing*. Some similarity measures were derived from distance measures using the formula: similarity = 1 − distance. The types of attribute-matching functions we used included:

Table 1
Attribute-matching functions

Matching function	Attribute pair		Description
	Catalog A	Catalog B	
AU	AUTHORS	AUTHOR	Equality comparison
AU2			Levenshtein's edit distance
TT	TITLE	TITLE	Equality comparison
TT2			Levenshtein's edit distance
LP	LIST_PRICE	RETAILPRICE	Equality comparison
LP2			$\mathrm{LP2}(x,y)=1-\frac{\lvert x-y\rvert}{x+y}$
LP3			$\mathrm{LP3}(x,y)=1-\frac{\lvert x-y\rvert}{\max(x,y)}$
OP	OUR_PRICE	OURPRICE	Equality comparison
OP2			$\mathrm{OP2}(x,y)=1-\frac{\lvert x-y\rvert}{x+y}$
OP3			$\mathrm{OP3}(x,y)=1-\frac{\lvert x-y\rvert}{\max(x,y)}$
COV	COVER	COVERFORMAT	Equality comparison + dictionary lookup
PG	PAGES	PAGES	Equality comparison
PG2			$\mathrm{PG2}(x,y)=1-\frac{\lvert x-y\rvert}{x+y}$
PG3			$\mathrm{PG3}(x,y)=1-\frac{\lvert x-y\rvert}{\max(x,y)}$
ED	EDITION	EDITION	Equality comparison
PM	MONTH	PUBMONTH	Equality comparison
PY	YEAR	PUBYEAR	Equality comparison
PD	MONTH + YEAR	PUBMONTH + PUBYEAR	$\mathrm{PD}((m1,y1),(m2,y2))=\min(1,1-\lvert(y1-y2)+(m1-m2)/12\rvert)$
PUB	PUBLISHER	PUBLISHER	Equality comparison + dictionary lookup
SRK	SALES_RANK	SALESRANK	$\mathrm{SRK}(x,y)=1-\frac{\lvert x-y\rvert}{x+y}$
RAT	AVG_RATING	RATING	$\mathrm{RAT}(x,y)=1-\frac{\lvert x-y\rvert}{x+y}$

- *Equality comparison*: Equality comparison is the simplest matching method. It simply returns 1 if the two values under comparison are equal and 0 otherwise. Some attribute pairs, such as edition (A.Edition and B.Edition), publishing month (A.Month and B.Pubmonth), and publishing year (A.Year and B.Pubyear) were required to match exactly for two records to be about the same book. Although any attribute pairs could be compared exactly, equality comparison might not be the best matching function. Table 2 summarizes the equality comparison results for each common attribute pair on the match tuple pairs. Most of the common attribute pairs, except publishing year and cover (A.Cover and B.Coverformat), matched for less than 55% of the match tuple pairs, showing that we needed more complex attribute-matching functions for some common attribute pairs.
- *Approximate string comparison*: There were many typographical discrepancies in author names (A.Authors and B.Author) and titles (A.Title and B.Titles). We used the edit distance metric defined by Levenshtein [43] to compute the distance between two strings. Levenshtein's edit distance between two strings is defined as the minimum number of characters that need to be inserted into or deleted from one string to transform it

Table 2
Statistics of the match tuple pairs on common attribute pairs

Attribute pair		Equal %	Not equal %	One missing %	Both missing %
Catalog A	Catalog B				
AUTHORS	AUTHOR	52.14	46.01	1.71	0.14
TITLE	TITLE	46.30	53.70	0.00	0.00
LIST_PRICE	RETAILPRICE	41.45	2.71	23.08	32.76
OUR_PRICE	OURPRICE	52.85	31.77	14.25	1.14
COVER	COVERFORMAT	75.36	20.66	3.99	0.00
PAGES	PAGES	37.32	39.74	16.24	6.70
EDITION	EDITION	17.09	1.57	31.20	50.14
MONTH	PUBMONTH	44.59	51.42	3.99	0.00
YEAR	PUBYEAR	83.19	12.82	3.99	0.00
PUBLISHER	PUBLISHER	20.66	73.50	5.84	0.00
SALES_RANK	SALESRANK	0.00	74.22	21.65	4.13
AVG_RATING	RATING	5.98	17.66	56.41	19.94

into another string divided by the total number of characters in the two strings. The degree of similarity between two strings was defined as one minus Levenshtein's edit distance.

- *Dictionary*: Dictionaries helped resolve coding differences. Most of the publishers were named differently in the two catalogs. For example, one publisher was named "John Wiley & Sons" in Catalog A and "Wiley, John & Sons, Incorporated" in Catalog B. Cover types were sometimes coded differently, too. For example, the cover of some books was coded as "Paperback" in Catalog A and "Textbook Paperback" in Catalog B. We built small dictionaries and used dictionary lookup, in addition to equality comparison, for these two common attribute pairs. The matching function for such an attribute pair returned 1 if two attribute values are equal or the pair exists in the dictionary built for the attribute pair and 0 otherwise.
- *Normalized numeric distance*: Some attribute pairs, such as list price (A.List_price and B.Retailprice), our price (A.Our_price and B.Ourprice), and number of pages (A.Pages and B.Pages), of the same book were different, but deemed to be close. We normalized the difference between two numbers into the range of $[0, 1]$. Sales rank (A.Sales_rank and B.Salesrank) and average rating (A.Avg_rating and B.Rating) were also compared based on normalized numeric distance functions. We also combined publishing month and publishing year into a single number and computed a normalized distance between two such numbers.

For some attribute pairs, such as author and title, we designed multiple matching functions (equality comparison and approximate string comparison). List price and our price were also compared both exactly and approximately (using two normalized distance functions). For a pair of common attributes, however, multiple matching functions are usually highly correlated. Table 3 shows the Pearson correlation coefficient between each pair of matching functions. Many of the different matching functions for the same common attribute pair were highly correlated, with a coefficient of determination, r^2, over 0.5, or a Pearson correlation coefficient, r, over 0.71. Some classification techniques, such as Fisher's linear discriminant analysis and logistic regression, are sensitive to highly correlated features in the training data. Using multiple matching functions for a pair of common attributes also biases the importance of the pair, especially in techniques such as naive Bayes, where the effects (i.e., posterior probabilities of particular attribute-matching function values given that a pair of tuples match or do not match) of these matching functions will be multiplied.

We selected one matching function for each pair of common attributes. Table 4 shows the Pearson correlation between each matching function and the match label. For a pair of common attributes, the matching function most highly correlated with the match label was used in the subsequent classification. It turned out that approximate comparison was better for some attribute pairs (author and title) while equality comparison was better for others (list price, our price, and number of pages).

We used several widely used classification methods available in Weka [49], including *1-rule* (1R), *logistic regression* (Logistic), *classification via linear regression* (Linear), *J4.8 decision tree* (J4.8), *decision table* (DecTab), *naive Bayes* (Bayes), *back propagation neural network* (BP), and *k-Nearest Neighbor* (1-NN, 3-NN, and

Table 3
Pearson correlation matrix of attribute-matching functions

	AU	AU2	TT	TT2	LP	LP2	LP3	OP	OP2	OP3	COV	PG	PG2	PG3	ED	PM	PY	PD	PUB	SRK
AU2	**0.85**																			
TT	0.32	0.46																		
TT2	0.55	**0.77**	0.67																	
LP	0.27	0.39	0.35	0.45																
LP2	0.12	0.19	0.20	0.23	**0.79**															
LP3	0.15	0.23	0.23	0.27	**0.83**	**0.99**														
OP	0.34	0.45	0.35	0.52	0.70	0.49	0.53													
OP2	0.17	0.27	0.21	0.30	0.42	0.46	0.48	0.51												
OP3	0.23	0.35	0.26	0.39	0.50	0.50	0.52	0.61	**0.98**											
COV	0.20	0.24	0.17	0.25	0.24	0.27	0.27	0.26	0.37	0.37										
PG	0.28	0.39	0.29	0.44	0.28	0.17	0.19	0.32	0.22	0.26	0.20									
PG2	0.14	0.25	0.23	0.31	0.41	0.43	0.45	0.34	0.47	0.49	0.37	0.46								
PG3	0.20	0.33	0.27	0.39	0.45	0.43	0.45	0.39	0.48	0.51	0.37	0.53	**0.99**							
ED	0.10	0.16	0.16	0.19	0.17	0.14	0.15	0.14	0.19	0.20	0.12	0.10	0.21	0.22						
PM	0.21	0.32	0.26	0.38	0.19	0.09	0.11	0.24	0.17	0.21	0.12	0.22	0.21	0.24	0.12					
PY	0.38	0.55	0.38	0.62	0.37	0.24	0.26	0.39	0.32	0.37	0.25	0.35	0.34	0.39	0.15	0.44				
PD	0.44	0.62	0.42	0.69	0.42	0.26	0.29	0.44	0.37	0.43	0.28	0.39	0.37	0.43	0.18	0.52	**0.88**			
PUB	0.42	0.64	0.47	**0.75**	0.46	0.28	0.31	0.49	0.34	0.42	0.28	0.41	0.37	0.44	0.19	0.39	0.60	0.65		
SRK	0.11	0.18	0.12	0.20	0.34	0.37	0.37	0.27	0.38	0.37	0.15	0.10	0.27	0.27	0.09	0.09	0.19	0.22	0.19	
RAT	−0.02	0.04	0.10	0.07	0.18	0.23	0.23	0.06	0.11	0.10	0.09	0.02	0.12	0.12	0.10	0.03	0.09	0.09	0.09	0.14

Correlation coefficients over 0.71 are highlighted.

Table 4
Pearson correlation coefficients between attribute-matching functions and the match label

Function	Correlation
AU	0.594
AU2	**0.826**
TT	0.549
TT2	**0.916**
LP	**0.465**
LP2	0.224
LP3	0.267
OP	**0.547**
OP2	0.303
OP3	0.402
COV	**0.24**
PG	**0.473**
PG2	0.317
PG3	0.408
ED	**0.196**
PM	**0.409**
PY	**0.669**
PD	**0.739**
PUB	**0.788**
SRK	**0.211**
RAT	**0.064**

The best matching function for each attribute pair is highlighted.

5-NN), to classify tuple pairs. The training data set consisted of 2808 examples of tuple pairs, each of which was represented by 13 attribute-matching function values and a match label (1 for match and 0 for non-match).

1R selects the single most discriminating feature to make the classification decision. In this example, TT2 was selected. The classification rules were

IF TT2 $\geqslant$ 0.425, Match; Otherwise, Non-Match.

The model learned by Logistic was
IF $D_{\text{Logistic}} \geqslant 12.6201$, Match; Otherwise, Non-Match

where

$$\begin{aligned} D_{\text{Logistic}} = {} & 9.6725 \times \text{AU2} + 15.2296 \times \text{TT2} - 0.9825 \times \text{LP} + 2.6276 \times \text{OP} - 0.6146 \times \text{COV} + 4.0138 \\ & \times \text{PG} + 2.5171 \times \text{ED} - 0.7047 \times \text{PM} + 2.9043 \times \text{PY} + 2.0387 \times \text{PD} + 2.9073 \times \text{PUB} \\ & - 0.9294 \times \text{SRK} - 1.6748 \times \text{RAT} \end{aligned}$$

The model learned by Linear was
IF $D_{\text{Linear}} \geqslant 0.1416$, Match; Otherwise, Non-Match where

$$\begin{aligned} D_{\text{Linear}} = {} & 0.3228 \times \text{AU2} + 0.6639 \times \text{TT2} - 0.0273 \times \text{LP} + 0.0658 \times \text{OP} - 0.0424 \times \text{COV} + 0.0534 \times \text{PG} \\ & + 0.0174 \times \text{PY} + 0.1292 \times \text{PD} + 0.1561 \times \text{PUB} \end{aligned}$$

Fig. 2 shows the decision tree generated by J4.8, Weka's implementation of C4.5. DecTab selected features AU2, TT2, COV, PG, and PUB for the table lookup when a new example needed to be classified. Bayes estimated a collection of prior probabilities and posterior probabilities. BP learned a collection of weights in a network with one hidden layer. k-Nearest Neighbor methods simply memorized the training examples and classified each new example according to the majority class of its k nearest training examples.

We conducted experiments to compare the *accuracies* of different classification techniques. We ran each method 200 times; each time, 66% of the examples in the input data set were randomly re-sampled for training, and the remaining examples were set aside for testing. Table 5 summarizes the results of the experiment. Table 6 summarizes the *ANOVA test* of the accuracies of the classification techniques, which indicates that at least two of the classification techniques performed significantly differently in terms of accuracy on the significance level $\alpha = 0.05$ ($F(9, 1990) = 1072.500$, $p < 0.05$). Table 7 summarizes the result of a *Sheffé's post hoc test*. Six homogeneous (in terms of accuracy) subsets of classification techniques were recognized at the significance level $\alpha = 0.05$. 1R was the least accurate; Bayes and BP were the most accurate.

4.3. Evaluating schema-level correspondences

We further analyzed the relationships between attributes across the two catalogs using statistical analysis methods after instance-level correspondences had been identified in the previous classification step. We used

```
TT2 <= 0.43
|   AU2 <= 0.37
|   |   PUB <= 0: Non-Match (1345/2)
|   |   PUB > 0
|   |   |   PD <= 0.5: Non-Match (30)
|   |   |   PD > 0.5
|   |   |   |   TT2 <= 0.24: Non-Match (5)
|   |   |   |   TT2 > 0.24
|   |   |   |   |   SRK <= 0.42: Non-Match (6/2)
|   |   |   |   |   SRK > 0.42: Match (6)
|   AU2 > 0.37
|   |   PD <= 0: Non-Match (8/2)
|   |   PD > 0: Match (68/2)
TT2 > 0.43
|   AU2 <= 0.23
|   |   PD <= 0.25
|   |   |   TT2 <= 0.89
|   |   |   |   AU2 <= 0.12: Match (5/1)
|   |   |   |   AU2 > 0.12: Non-Match (12)
|   |   |   TT2 > 0.89: Match (8)
|   |   PD > 0.25: Match (59/1)
|   AU2 > 0.23: Match (1256)
```

Fig. 2. A J4.8 decision tree. (The two numbers attached to each leaf node are the total number of examples covered by the node and the number of examples incorrectly classified by the node in the training data.)

Table 5
Summary of classification results

Method	N	Accuracy (%)		False positive rate (%)		False negative rate (%)		Training time (s)		Testing time (s)	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
1R	200	96.581	0.524	1.650	0.884	5.190	0.970	0.11	0.05	0.02	0.03
3-NN	200	98.298	0.391	1.619	0.685	1.786	0.565	0.01	0.00	33.32	3.71
Linear	200	98.315	0.361	0.075	0.101	3.296	0.723	1.57	0.34	0.11	0.10
5-NN	200	98.326	0.368	1.368	0.549	1.981	0.591	0.01	0.01	33.72	3.07
J4.8	200	98.871	0.317	1.009	0.593	1.250	0.626	0.67	0.26	0.03	0.06
DecTab	200	99.009	0.327	1.047	0.655	0.936	0.570	4.25	0.52	0.06	0.04
Logistic	200	99.088	0.277	0.689	0.398	1.135	0.513	0.41	0.11	0.06	0.08
1-NN	200	99.130	0.350	0.880	0.461	0.861	0.563	0.01	0.01	24.70	1.91
Bayes	200	99.240	0.260	1.248	0.475	0.272	0.197	0.14	0.03	0.08	0.03
BP	200	99.306	0.271	0.586	0.356	0.804	0.471	210.45	46.44	0.21	0.08

Table 6
ANOVA of the accuracies of 10 classifiers

	Sum of squares	df	Mean square	F	Sig.
Between groups	1197.955	9	133.106	1072.500	0.000
Within groups	246.975	1990	0.124		
Total	1444.930	1999			

Table 7
Sheffé's test of the accuracies of 10 classifiers—homogeneous subsets

Method	N	Subset for alpha = .05					
		1	2	3	4	5	6
1R	200	96.581					
3-NN	200		98.298				
Linear	200		98.315				
5-NN	200		98.326				
J4.8	200			98.871			
DecTab	200			99.009	99.009		
Logistic	200				99.088		
1-NN	200				99.130	99.130	
Bayes	200					99.240	99.240
BP	200						99.306
Sig.		1.000	1.000	0.084	0.221	0.380	0.941

correlation analysis first to verify the correspondences between numeric attributes identified in the previous cluster analysis step and then to detect other potential correspondences between numeric attributes that previously had been missed. We brought match tuple pairs from the two catalogs into a unified data set and computed the Pearson correlation coefficient between every pair of numeric attributes across the two catalogs. Table 8 summarizes the Pearson correlation coefficients between the pairs of numeric attributes.

For four attribute pairs, including publishing month (A.Month and B.Pubmonth), publishing year (A.Year and B.Pubyear), sales rank (A.Sales_rank and B.Salesrank), and rating (A.Avg_rating and B.Rating), the two attributes in each pair had been clustered as similar in the previous cluster analysis step but were found not highly correlated. The coefficient of determination (r^2) between the two attributes in each of these pairs was below 0.5. The two catalogs did not agree well on publishing month and publishing year, primarily because of data errors. Another reason was that some books had not yet been published and the two catalogs used different predicted publishing dates. These were corresponding attribute pairs but had some discrepancies in their actual values. The two catalogs did not agree well on sales rank and rating because the two bookstores gen-

Table 8
Person correlation matrix of numeric attributes

		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
V1	A.LIST_PRICE															
V2	B.RETAILPRICE	**1.00**														
V3	A.OUR_PRICE	**1.00**	**0.97**													
V4	B.OURPRICE	**0.97**	**0.99**	**0.99**												
V5	A.EDITION	−0.15	−0.18	−0.07	−0.06											
V6	B.EDITION	0.03	−0.14	−0.05	0.01	**0.86**										
V7	A.PAGES	**0.71**	**0.69**	0.21	0.25	0.08	0.14									
V8	B.PAGES	**0.69**	**0.65**	0.15	0.17	0.07	0.13	**0.97**								
V9	A.MONTH	0.07	0.07	0.06	0.10	0.05	0.02	0.07	0.04							
V10	B.PUBMONTH	0.03	0.04	−0.02	−0.01	0.04	0.05	0.07	0.00	**0.57**						
V11	A.YEAR	0.16	0.16	0.04	0.04	−0.17	0.14	0.13	−0.02	−0.07	0.02					
V12	B.PUBYEAR	0.16	0.18	−0.03	−0.10	−0.13	0.15	0.13	0.00	−0.01	0.05	**0.51**				
V13	A.SALES_RANK	−0.12	−0.03	0.11	0.15	−0.07	−0.07	−0.23	−0.06	−0.01	−0.14	−0.12	−0.18			
V14	B.SALESRANK	−0.18	−0.11	−0.08	0.04	−0.13	−0.19	−0.29	−0.26	−0.01	−0.04	−0.19	−0.28	**0.51**		
V15	A.AVG_RATING	−0.11	−0.07	−0.09	−0.08	0.07	0.02	−0.10	−0.12	0.04	0.01	−0.07	−0.06	0.06	0.06	
V16	B.RATING	−0.27	−0.23	−0.27	−0.22	0.23	0.24	−0.11	−0.12	0.08	0.00	−0.08	−0.06	0.06	−0.18	**0.57**

Correlation coefficients over 0.5 are highlighted.

erated these numbers independently. Thus, these attribute pairs were not really corresponding ones, and the correlation analysis effectively revealed such differences across the two catalogs on these attribute pairs.

All other numeric attribute pairs that had previously been clustered as similar indeed appeared to be highly correlated. The correlation analysis therefore confirmed these attribute pairs as candidate corresponding ones.

The correlation analysis also detected some highly correlated pairs of attributes that had not been identified previously in the cluster analysis step. All four attributes A.List_price, B.Retailprice, A.Our_price, and B.Ourprice were highly correlated ($r \geqslant 0.95$). A.Pages and B.Pages were moderately correlated to A.Listprice and B.Retailprice ($r \geqslant 0.65$). Highly correlated attributes, however, might not semantically correspond to the same property about some real world entity type. For example, A.List_price and A.Our_price were correlated but described different properties about books. The correlation analysis revealed such related but not directly corresponding (i.e., describing the same property of some entity type) attributes.

We then used regression analysis to further analyze the relationships between correlated attributes. Note that the distinction between dependent variable and independent variable is arbitrary here; two attributes that are considered common to different databases are symmetric in the regression model. For a pair of attributes under analysis, we ran linear regression twice, using each attribute on different sides of the model. Table 9 shows some of the linear models about common attribute pairs. For the attribute pairs identified in the previous cluster analysis, the coefficients of the attributes in the models were very close to 1, indicating that these pairs were very likely to be common attribute pairs with no scaling differences. For the attribute pairs just discovered, the coefficients of the attributes in the models were not very close to 1, indicating that these pairs were not describing common properties or had scaling differences.

The coefficient of determination, R^2, of a linear regression model indicates the strength of the relationship between the independent variable and the dependent variable. Some of the models had high coefficients of determination, $R^2 \geqslant 0.9$, indicating strong relationships between the attributes. The relationships between list prices and numbers of pages were weak: $R^2 \leqslant 0.55$.

To verify the correspondences between character attributes identified in the previous cluster analysis step and to detect other potential correspondences between character attributes that had been missed previously, we used some attribute-matching functions, such as equality comparison, edit distance, and dictionary lookup, and observed the degree of correlation between the attribute-matching functions and the match label in a data set containing both match examples and non-match examples. When we used an attribute-matching function TYP, which combined equality comparison and dictionary lookup to compare A.Type and B.Editiondesc, the Pearson correlation coefficient between TYP and the match label was relatively high (=0.819), indicating that the two attributes potentially referred to the same property of books. The two attributes were indeed seman-

Table 9
Linear regression models between semantically-related attributes

Linear regression model	R^2
A.List_price = B.Retailprice × 0.997 + 0.059	0.994
B.Retailprice = A.List_price × 0.997 + 0.185	0.994
A.Our_price = B.Ourprice × 0.987 − 0.317	0.980
B.Ourprice = A.Our_price × 0.994 + 0.995	0.980
A.Pages = B.Pages × 0.955 + 22.283	0.933
B.Pages = A.Pages × 0.977 + 11.432	0.933
A.List_price = B.Ourprice × 1.133 + 1.354	0.940
A.List_price = A.Our_price × 1.229 + 0.336	0.992
B.Retailprice = A.Our_price × 1.149 + 1.775	0.946
B.Retailprice = B.Ourprice × 1.261 − 0.870	0.981
A.List_price = B.Pages × 0.035 + 18.786	0.472
A.List_price = A.Pages × 0.035 + 18.395	0.502
B.Retailprice = A.Pages × 0.035 + 18.634	0.470
B.Retailprice = B.Pages × 0.033 + 19.812	0.417

tically similar; they both described whether a CD or Disk is included in a book. The correspondence between A.Type and B.Editiondesc was therefore detected by this further analysis.

The analysis improved the schema-level correspondences identified in the previous cluster analysis step. It revealed that some attribute pairs, such as sales rank and average rating, which had been identified as similar in the previous cluster analysis step, were actually not so similar. Other attribute pairs that had been identified as similar in the previous cluster analysis step were further confirmed. Some new attribute correspondences, such as book type in catalog A and edition description in catalog B, which had not been identified in the previous cluster analysis step, were caught in the exploration of potential attribute correspondences. Such improvement in schema-level correspondences triggered a new iteration of the iterative procedure, described in the next subsection.

4.4. Further iterations

Based on an improved understanding of schema-level correspondences, we examined instance-level correspondences again, using the classification techniques we have described in Section 4.2. We conducted another experiment to see whether classification results would be significantly improved. If they were, we would evaluate schema-level correspondences again on an improved integrated data set and keep analyzing correspondences on the two levels alternately until no significant improvement could be observed.

We used the same examples in the input data we had used previously. We ran the same 10 classification techniques on 12 attribute-matching functions; two functions, SRK and RAT, were dropped; a new function, TYP, was added. Each technique was run 200 times; each time, 66% of the examples in the input data set were randomly selected for training while the remaining examples were set aside for testing. Table 10 summarizes the experiment results. Table 11 summarizes a series of *t*-tests that compare the accuracy of a classifier in the second iteration with that in the first iteration. No classifier was significantly improved ($p > 0.05$). Our improved understanding of schema-level correspondences (dropping SRK and RAT and adding TYP) did not significantly improve our detecting of instance-level correspondences. We therefore terminated the iterative procedure after the second iteration.

4.5. Another example

We have evaluated the proposed methodology and techniques in another application, one where the databases of two departments of a large public university, the Property Management Department and the Surplus Property Office, needed to be integrated. The Property Management Department manages the property assets owned by various departments of the university and keeps a record of every property asset in its database. The Surplus Property Office is responsible for disposing of unwanted property items and keeps a record of every item it has received for disposal. Because the two databases, named FFX and Surplus, respectively, were independently developed by different people at different times for different purposes, not surprisingly, they are very

Table 10
Summary of classification results in the second iteration

Base method	N	Accuracy (%)		False positive rate (%)		False negative rate (%)		Training time (s)		Testing time (s)	
		Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
1R	200	96.581	0.524	1.650	0.884	5.190	0.970	0.11	0.05	0.02	0.01
3-NN	200	98.281	0.413	1.636	0.673	1.800	0.639	0.01	0.01	31.37	5.17
Linear	200	98.317	0.361	0.075	0.101	3.292	0.723	1.27	0.19	0.09	0.08
5-NN	200	98.364	0.374	1.314	0.553	1.957	0.599	0.01	0.00	28.84	1.22
J4.8	200	98.840	0.295	0.997	0.616	1.324	0.603	0.42	0.08	0.02	0.03
DecTab	200	99.001	0.326	1.037	0.670	0.963	0.572	3.86	0.45	0.08	0.07
Logistic	200	99.136	0.275	0.586	0.360	1.143	0.475	0.38	0.10	0.06	0.08
1-NN	200	99.032	0.360	0.935	0.459	1.000	0.582	0.02	0.03	21.40	0.77
Bayes	200	99.240	0.256	1.238	0.450	0.283	0.214	0.12	0.12	0.07	0.01
BP	200	99.260	0.284	0.681	0.422	0.800	0.459	188.03	8.44	0.19	0.08

Table 11
t-Tests (df = 398)—improvement of classification accuracy in the second iteration

Base method	N	First iteration		Second iteration		t	p
		Mean	StdDev	Mean	StdDev		
1R	200	96.581	0.524	96.581	0.524	0.000	1.000
3-NN	200	98.298	0.391	98.281	0.413	−0.443	0.329
Linear	200	98.315	0.361	98.317	0.361	0.058	0.954
5-NN	200	98.326	0.368	98.364	0.374	1.016	0.155
J4.8	200	98.871	0.317	98.840	0.295	−1.008	0.314
DecTab	200	99.009	0.327	99.001	0.326	−0.257	0.797
Logistic	200	99.088	0.277	99.136	0.275	1.727	0.085
1-NN	200	99.130	0.350	99.032	0.360	−2.757	0.003
Bayes	200	99.240	0.260	99.240	0.256	0.000	1.000
BP	200	99.306	0.271	99.260	0.284	−1.641	0.051

different in both their schemas and instance data. Our task was to identify the corresponding attributes and records across the two databases.

We focused on the parts of the two databases that overlap. A table named INVMSTR in Surplus is closely related to several tables in FFX; both store data about property items. Hereafter, we refer to the join of the several tables in FFX as FFX. There are 115 and 32 attributes in FFX and INVMSTR, respectively. There are 77,966 and 13,365 records in the snapshots we took from FFX and INVMSTR, respectively. Due to the space limit, we will not present the detailed results, which can be found in [51].

When preparing features for clustering attributes, we found that only data patterns are easily available; there are serious problems with other possible features. Almost all attribute names are abbreviations and do not match across the two tables. Surplus does not have an online document describing its database. The data types and other schema specifications are incompatible between the two systems managing the databases, IBM IDMS and Foxpro, respectively. We therefore selected the same 14 data patterns and ran the same clustering methods as in the bookstore example. Since only data patterns were used, the clustering results were rougher than those in the bookstore example. The clusters were bigger, with more unrelated attributes incorrectly clustered as similar. On the other hand, seven of the 12 corresponding attribute pairs were not clustered as similar.

Next, we skipped an iteration of instance-level correspondence identification and directly analyzed attribute correspondences using statistical analysis techniques. In this example, there is again a common key, which is a tag number uniquely identifying each property item. Using a sample with 3124 matching record pairs, we performed correlation and regression analysis to verify the attribute correspondences identified in the previous cluster analysis and to detect potential correspondences between numeric attributes that had been previously missed. In another sample with 3124 matching record pairs and 3124 non-matching record pairs, we defined some attribute-matching functions, such as exact comparison, edit distance, and dictionary lookup, for each pair of character attributes and observed the degree of correlation between the attribute-matching functions and the match label. All the potential attribute correspondences identified by the statistical analyses were presented to the domain expert, who confirmed 12 corresponding attribute pairs.

We then evaluated the same classification techniques for classifying record pairs as in the bookstore example. The sample used in analyzing correspondences between character attributes was used again for training classifiers. The same attribute-matching functions for character attributes used in the statistical analysis were used again as input features for classification. Numeric attributes were compared using normalized numeric distances. The top three classifiers learned by J4.8, Logistic, and BP had estimated accuracy of 97.005%, 96.984%, and 96.944%, respectively. Another iteration did not lead to significant improvement in identified correspondences; hence, the procedure was terminated.

5. Conclusion and future work

Data integration is needed in many situations. In this paper, we have described a multilevel, multitechnique approach to identifying semantic correspondences from heterogeneous databases, which is a critical step in integrating the databases. While previous research has studied the identification of semantic correspondences on either the schema level or the instance level, we have proposed an iterative procedure in which semantic correspondences on the two levels are examined alternately and incrementally. We use cluster analysis techniques to initially identify some similar schema elements (i.e., relations and attributes). We then apply classification techniques to identify matching tuples from known semantically corresponding relations, based on known semantically corresponding attributes. These matching tuples are combined into a single data set, on which statistical analysis techniques are used to further analyze the relationships among relations and schemas. Improvement in the understanding of schema-level correspondences triggers another iteration of the procedure. This iterative procedure reduces both the amount of human involvement and the requirement for a particular automated technique. We have conducted empirical evaluations using real-world data to demonstrate the utility of the proposed approach.

We have reviewed various cluster analysis techniques developed in both multivariate statistical analysis and artificial neural networks and evaluated their use in the context of clustering schema elements from heterogeneous databases. We have also developed an SOM application for clustering schema elements. As past studies have indicated that no clustering method is superior to others in all situations [1,12], we encourage users to run multiple methods and compare the results. We also urge users to evaluate the cluster analysis results carefully in light of domain knowledge. While we did not find significant difference among different clustering methods in terms of accuracy, SOM intuitively appeared better than other methods for visualizing the results and allowing users to incrementally evaluate the candidate solutions. Users can start with the most similar attributes and gradually examine less similar ones. We therefore recommend users to use the SOM application as the primary tool and use other clustering methods as secondary references. If multiple methods agree on some clusters, it gives users more confidence on the validity of these clusters. Otherwise, users should pay more attention to the conflicting parts.

We have reviewed various classification techniques drawn from multivariate statistical analysis, data mining, and artificial neural networks and have evaluated their use in the context of matching tuples from heterogeneous databases. Since the performance of classification techniques varies across situations, we encourage users to experiment with various techniques in each particular situation to select the most appropriate one. The "no-free-lunch theorems" [50] have proven that there is no universally superior learning technique that best suits every application. Numerous empirical studies (e.g., [24,46]) in different applications have also found that no single learning technique can perform the best in all applications. Therefore, an "empirical" approach, instead of a prescription of panaceas, is usually recommended. Upon a particular application, the user needs to empirically evaluate several promising techniques—if the available resources permit—to determine the most appropriate one to use. We do not recommend any panacea either and indeed believe such a panacea does not exist.

We have reviewed statistical analysis techniques, such as correlation and regression, which can be used to analyze relationships among attributes after data from heterogeneous databases have been partially integrated. These techniques can be used to verify previously identified attribute correspondences, to explore potentially missed correspondences, and to test hypothetical correspondences. As highly correlated attributes do not necessarily correspond with the same property about some real-world entity type, we encourage users to evaluate the findings in light of domain knowledge.

While we have proposed and evaluated a comprehensive, iterative procedure for identifying both schema-level and instance-level semantic correspondences from heterogeneous databases, we are not suggesting that all of the steps in the procedure should be followed strictly in every data integration project. For example, if the schemas are relatively small—such as the schemas of the book catalogs used as an illustrative example in this paper—they may be manually compared and integrated, without using any of the automated techniques described in this paper. However, if the schemas are very large—such as the schemas of the US Air Force databases described by Clifton et al. [8]—automated support is critical to reducing the amount of human time and effort. If the databases do not share any common records—for example, if two corporations are merged and need to integrate their human resource databases that do not contain any common employees—the step for identifying corresponding records is not needed at all. If the databases all use a common key—for example, if several healthcare related databases maintained by different hospitals, clinics, and healthcare insurance companies all strictly use the Social Security Number as the key for patients—the patient records can be easily matched across databases using the common key, without training any classifiers. The general procedure proposed in this paper should be customized for different applications based on the particular complexity, characteristics, and requirements of the applications.

As part of our future research, we would like to conduct more empirical evaluation in real-world data integration situations to gain insights about the generalizability of our overall approach and particular techniques. We hope to produce some heuristics or guidelines that can help practitioners in tailoring the procedure and selecting appropriate techniques according to their particular situations.

Another area of future work is the development of a system that streamlines the techniques involved in the procedure. The system should be able to access various types of databases, such as relational databases, semi-structured databases, and text files. The system should include the following components: (1) tools for extracting semantic features about schema elements, e.g., a thesaurus for comparing the names of schema elements, information retrieval tools for comparing documents, and templates for querying statistics about data patterns; (2) dimensionality reduction and cluster analysis tools; (3) data-cleaning-and-standardization, data-transformation, and attribute-matching methods; (4) classification tools for identifying matching tuples; (5) statistical analysis tools for analyzing schema-level correspondences; and (6) tools that allow users to view, add, confirm, reject, and modify semantic correspondences.

Finally, our approach to identifying semantic correspondences can be incorporated into a complete framework for heterogeneous database integration. Our approach is applicable in both logical (e.g., federated schema approach) and physical (e.g., data warehousing) integration strategies. Schema-level semantic correspondences are the basis for schema integration. Instance-level semantic correspondences are the basis for data integration and duplicate elimination.

Acknowledgements

The authors are grateful to the anonymous referees for their valuable suggestions.

References

[1] A.A. Afifi, V. Clark, Computer-Aided Multivariate Analysis, third ed., Chapman & Hall, 1996.

[2] A.P. Ambrosio, E. Métais, J. Meunier, The linguistic level: contribution for conceptual design, view integration, reuse and documentation, Data & Knowledge Engineering 21 (2) (1997) 111–129.

[3] G.B. Bell, A. Sethi, Matching records in a national medical patient index, Communications of the ACM 44 (8) (2001) 83–88.

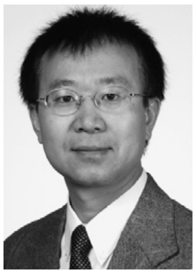
[4] S.S. Benkley, J.F. Fandozzi, E.M. Housman, G.M. Woodhouse, Data element tool-based analysis (DELTA), The MITRE Corporation, Bedford, MA, Technical Report, MTR 95B0000147, 1995.

[5] M.W. Bright, A.R. Hurson, S.H. Pakzad, Automated resolution of semantic heterogeneity in multidatabases, ACM Transactions on Database Systems 19 (2) (1994) 212–253.

[6] C.D. Budzinsky, Automated spelling correction, Statistics Canada, Unpublished Report, 1991.

[7] A.L.P. Chen, P.S.M. Tsai, J.L. Koh, Identifying object isomerism in multidatabase systems, Distributed and Parallel Databases 4 (2) (1996) 143–168.
[8] C. Clifton, E. Housman, A. Rosenthal, Experience with a combined approach to attribute-matching across heterogeneous databases, in: Data Mining and Reverse Engineering, Proceedings of DS-7, 1997, pp. 429–451.
[9] J.A.F. Costa, M.L. de Andrade Netto, Estimating the number of clusters in multivariate data by self-organizing maps, International Journal of Neural Systems 9 (3) (1999) 195–202.
[10] D. Dey, S. Sarkar, P. De, A probabilistic decision model for entity matching in heterogeneous databases, Management Science 44 (10) (1998) 1379–1395.
[11] E. Ellmer, C. Huemer, D. Merkl, G. Pernul, Automatic classification of semantic concepts in view specifications, in: Database and Expert Systems Applications, Proceedings of 7th International Conference, DEXA '96, 1996, pp. 824–833.
[12] B.S. Everitt, S. Landau, M. Leese, Cluster Analysis, fourth ed., Arnold, London, 2001.
[13] M.E. Fair, Record linkage in an information age society, in: Record Linkage Techniques—1997, 1997, pp. 427–441.
[14] W. Fan, H. Lu, S.E. Madnick, D. Cheung, Discovering and reconciling value conflicts for numerical data integration, Information Systems (26) (2001) 635–656.
[15] I.P. Fellegi, A.B. Sunter, A theory of record linkage, Journal of the American Statistical Association 64 (328) (1969) 1183–1210.
[16] M. Ganesh, J. Srivastava, T. Richardson, Mining entity-identification rules for database integration, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996, pp. 291–294.
[17] I.J. Haimowitz, Ö. Gür-Ali, H. Schwarz, Integrating and mining distributed customer databases, in: Proceedings of KDD-97, 1997, pp. 179–182.
[18] S. Hayne, S. Ram, Multi-user view integration system (MUVIS): an expert system for view integration, in: Proceedings of the Sixth International Conference on Data Engineering, 1990, pp. 402–410.
[19] M.A. Hernández, S.J. Stolfo, Real-world data is dirty: data cleansing and the merge/purge problem, Data Mining and Knowledge Discovery 2 (1) (1998) 9–37.
[20] P. Johannesson, Supporting schema integration by linguistic instruments, Data & Knowledge Engineering 21 (2) (1997) 165–182.
[21] R.E. Kirk, Statistics: An Introduction, fourth ed., Harcourt Brace College Publishers, 1999.
[22] T. Kohonen, Self-Organizing Maps, Springer, Berlin, 1995.
[23] W.S. Li, C. Clifton, SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks, Data & Knowledge Engineering 33 (1) (2000) 49–84.
[24] T.S. Lim, W.Y. Loh, Y.S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning Journal (40) (1998) 203–228.
[25] H. Lu, W. Fan, C.H. Goh, S.E. Madnick, D.W. Cheng, Discovering and reconciling semantic conflicts: a data mining perspective, in: Data Mining and Reverse Engineering, Proceedings of DS-7, 1997, pp. 410–427.
[26] S. Luján-Mora, M. Palomar, Reducing inconsistency in integrating data from different sources, in: Proceedings of the 2001 International Database Engineering and Applications Symposium, 2001, pp. 209–218.
[27] J. Madhavan, P.A. Bernstein, E. Rahm, Generic schema matching with cupid, in: Proceedings of the 27th International Conferences on Very Large Databases, 2001, pp. 49–58.
[28] P. Mangiameli, S.K. Chen, D. West, A comparison of SOM neural network and hierarchical clustering methods, European Journal of Operational Research 93 (2) (1996) 402–417.
[29] N. Masood, B. Eaglestone, Semantics based schema analysis, in: Database and Expert Systems Applications, 9th International Conference (DEXA'98), 1998, pp. 80–89.
[30] I. Mirbel, Semantic integration of conceptual schemas, Data & Knowledge Engineering 21 (2) (1997) 183–195.
[31] S. Navathe, H. Thomas, M. Satitsamitpong, A. Datta, A model to support E-catalog integration, in: Semantic Issues in e-Commerce Systems, Proceedings of DS-9, 2001, pp. 247–261.
[32] L. Palopoli, L. Pontieri, G. Terracina, D. Ursino, Intensional and extensional integration and abstraction of heterogeneous databases, Data & Knowledge Engineering 35 (3) (2000) 201–237.
[33] H. Petersohn, Assessment of cluster analysis and self-organizing maps, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (2) (1998) 136–149.
[34] J.C. Pinheiro, D.X. Sun, Methods for linking and mining massive heterogeneous databases, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 1998, pp. 309–313.
[35] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, San Mateo, CA, 1993.
[36] S. Ram, R. Venkataraman, Schema integration: past, present and future, in: A. Elmagarmid, M. Rusinkiewicz, A. Sheth (Eds.), Management of Heterogeneous and Autonomous Database Systems, Morgan Kaufman Publishers, 1999, pp. 119–156.
[37] S. Ram, H. Zhao, Detecting both schema-level and instance-level correspondences for the integration of E-catalogs, in: Proceedings of the Eleventh Annual Workshop on Information Technology and Systems (WITS'01), New Orleans, Louisiana, USA, 2001, pp. 193–198.
[38] M.A. Rodríguez, M.J. Egenhofer, R.D. Rugg, Assessing semantic similarities among geospatial feature class definitions, in: Proceedings of the Second International Conference on Interoperating Geographic Information Systems (INTEROP'99), Lecture Notes in Computer Science, vol. 1580, 1999, pp. 189–202.
[39] A. Segev, A. Chatterjee, A framework for object matching in federated databases and its implementation, International Journal of Cooperative Information Systems 5 (1) (1996) 73–99.
[40] A. Si, C.C. Ying, D. McLeod, On using historical update information for instance identification in federated databases, in: Proceedings of the First IFCIS International Conference on Cooperative Information Systems, 1996, pp. 68–77.

[41] W.W. Song, P. Johannesson, J.A. Bubenko, Semantic similarity relations and computation in schema integration, Data & Knowledge Engineering 19 (1) (1996) 65–97.
[42] U. Srinivasan, A.H.H. Ngu, T. Gedeon, Managing heterogeneous information systems through discovery and retrieval of generic concepts, Journal of the American Society for Information Science 51 (8) (2000) 707–723.
[43] G.A. Stephen, String Searching Algorithms, World Scientific Publishing Co. Pte. Ltd., 1994.
[44] S. Tejada, C.A. Knoblock, S. Minton, Learning object identification rules for information integration, Information Systems 26 (8) (2001) 607–633.
[45] V.S. Verykios, A.K. Elmagarmid, E.N. Houstis, Automating the approximate record-matching process, Information Sciences 126 (1–4) (2000) 83–98.
[46] S.M. Weiss, C.A. Kulikowski, Computer Systems That Learn—Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert System, Morgan Kaufman Publishers, Inc., 1991.
[47] W.E. Winkler, Matching and record linkage, in: Record Linkage Techniques—1997, 1997, pp. 374–403.
[48] W.E. Winkler, Record linkage software and methods for merging administrative lists, in: Exchange of Technology and Know-How, Eurostat, Luxembourg, 1999, pp. 313–323.
[49] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufman Publishers, San Francisco, California, 2005.
[50] D.H. Wolpert, The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework, in: D.H. Wolpert (Ed.), The Mathematics of Generalization—Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning, Addison-Wesley, 1994, pp. 117–214.
[51] H. Zhao, Combining schema and instance information for integrating heterogeneous databases: an analytical approach and empirical evaluation, Ph.D. dissertation, University of Arizona, 2002.
[52] H. Zhao, S. Ram, Clustering schema elements for semantic integration of heterogeneous data sources, Journal of Database Management 15 (4) (2004) 88–106.
[53] H. Zhao, S. Ram, Entity identification for heterogeneous database integration—a multiple classifier system approach and empirical evaluation, Information Systems 30 (2) (2005) 119–132.
[54] H. Zhao, S. Ram, Clustering similar schema elements across heterogeneous databases: a first step in database integration, in: K. Siau (Ed.), Advanced Topics in Database Research, vol. 5, Idea Group Publishing, 2006, pp. 235–256.



Huimin Zhao is an Assistant Professor of Management Information Systems at the Sheldon B. Lubar School of Business, University of Wisconsin-Milwaukee. He earned his Ph.D. in Management Information Systems from the University of Arizona. His current research interests are in the areas of data integration, data mining, and Web services. His research has been published in several journals, including IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Systems, Man, and Cybernetics, Information Systems, Journal of Management Information Systems, Journal of Database Management, Journal of Information Systems and e-Business Management, and International Journal of Web Services Research. He is a member of IEEE, AIS, and IRMA.

Sudha Ram is Eller Professor of Management Information Systems in the Eller College of Business and Public Administration at the University of Arizona. She received a B.S. Degree in Mathematics, Physics and Chemistry from the University of Madras in 1979, PGDM from the Indian Institute of Management, Calcutta in 1981, and a Ph.D. from the University of Illinois at Urbana-Champaign, in 1985. Dr. Ram has published articles in such journals as Communications of the ACM, IEEE Expert, IEEE Transactions on Knowledge and Data Engineering, Information Systems, Information Systems Research, Management Science, and MIS Quarterly. Her research deals with issues related to Enterprise Data Management. Her research has been funded by organizations such as, IBM, Intel Corporation, Raytheon, US ARMY, NIST, NSF, NASA, and Office of Research and Development of the CIA. Specifically, her research deals with Interoperability among Heterogeneous Database Systems, Semantic Modeling, BioInformatics and Spatio-Temporal Semantics, Business Rules Modeling, Web services Discovery and Selection, and Automated software tools for database design. Dr. Ram serves on editorial board for such journals as Decision Support Systems, Information Systems Frontiers, Journal of Information Technology and Management, and as associate editor for Information Systems Research, Journal of Database Management, and the Journal of Systems and Software. She has chaired several workshops and conferences supported by ACM, IEEE, and AIS. She is a cofounder of the Workshop on Information Technology and Systems (WITS) and serves on the steering committee of many workshops and conferences including the Entity Relationship Conference (ER). Dr. Ram is a member of ACM, IEEE Computer Society, INFORMS, and Association for Information Systems (AIS). She is also the director of the Advanced Database Research Group based at the University of Arizona.