Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Systems 30 (2005) 119–132

www.elsevier.com/locate/infosys

Entity identification for heterogeneous database integration—a multiple classifier system approach and empirical evaluation

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

[a] School of Business Administration, University of Wisconsin, Milwaukee, P.O. Box 742, Milwaukee, WI 53201, USA
[b] Department of Management Information Systems, University of Arizona, McClelland Hall 430, Tucson, AZ 85721, USA

Received 16 May 2003; received in revised form 21 October 2003; accepted 3 November 2003

Abstract

Entity identification, i.e., detecting semantically corresponding records from heterogeneous data sources, is a critical step in integrating the data sources. The objective of this research is to develop and evaluate a novel multiple classifier system approach that improves entity identification accuracy. We apply various classification techniques drawn from statistical pattern recognition, machine learning, and artificial neural networks to determine whether two records from different data sources represent the same real-world entity. We further employ a variety of ways to combine multiple classifiers for improved classification accuracy. In this paper, we report on some promising empirical results that demonstrate performance improvement by combining multiple classifiers.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Heterogeneous database integration; Entity identification; Multiple classifier system

1. Introduction

The need to integrate heterogeneous data sources is ubiquitous. Legacy databases developed over time in different sections of an organization need to be integrated for strategic purposes. Business mergers and acquisitions force information systems previously owned by different institutions to be merged. Information needs to be shared or exchanged across system boundaries of cooperating enterprises. The rapid growth of the Internet continuously amplifies the need for semantic interoperability across heterogeneous data sources. The numerous data sources on the World-Wide-Web create new requirements and opportunities for data integration.

In order to integrate a collection of heterogeneous data sources, either logically (e.g., using a mediator/wrapper architecture) or physically (e.g., building a data warehouse), a critical step is to identify semantically corresponding records, i.e., records that represent the same entity in the real world, from the data sources. This problem has been referred to as *entity identification* [1], *approximate record matching* [2], *merge/purge* [3], and *record linkage* [4,5]. It has been shown to be a very complex and time-consuming task due to dirty data and semantic heterogeneities among different data sources. Winkler reported that integration of several mailing lists

*Corresponding author. Tel.: +1-414-229-6524; fax: +1-414-229-5999.
E-mail addresses: hzhao@uwm.edu (H. Zhao), ram@bpa.arizona.edu (S. Ram).

0306-4379/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2003.11.001

for the US Census of Agriculture in 1992 consumed thousands of person-hours even though an automated matching tool was used [5].

In this paper, we propose a novel *multiple classifier system* approach to entity identification. We apply multiple classification techniques drawn from *statistical pattern recognition*, *machine learning*, and artificial *neural networks* to determine whether two records from different data sources represent the same real-world entity. While past research has been committed to a particular technique, such as *record linkage* [4,5], *logistic regression* [6], or *decision tree* [1,2,7,8], we conducted experiments to select the best techniques in each particular case, because the performance of classification techniques varies in different situations; there is no single technique that is always superior to others. We further combine multiple classifiers in a variety of ways for potential improvement of classification accuracy. We have empirically evaluated our approach using real-world heterogeneous data sources and report some promising experimental results in this paper.

The paper is organized as follows. In the next section, we briefly review some past approaches for entity identification. In Section 3, we discuss some widely used classification techniques that can be applied to entity identification and some attribute-matching functions that can be used as features for classification. In Section 4, we propose a multiple classifier system approach to improving classification accuracy and describe a variety of ways to combine multiple classifiers. We then report some empirical evaluation results in Section 5. Finally, we summarize the contributions of this work and discuss future research directions.

2. Related work

Various approaches, both *rule*- and *learning-based*, have been proposed for detecting semantically corresponding records from heterogeneous data sources. They differ in how they generate *decision rules* to establish the correspondence between two records.

Rule-based approaches elicit decision rules from domain experts. Most of these rules involve comparing an overall similarity score and a threshold value. The overall similarity score for two records is usually a linear combination of similarity degrees between common attribute values of the two records. In Chen et al.'s approach [9], the weight (i.e., coefficient in the linear model) of each common attribute is specified manually according to its importance in determining whether two records match or not. In Segev and Chatterjee's approach [10], the weights and thresholds are specified manually based on domain knowledge or estimated using some statistical techniques such as logistic regression. Dey et al. [11] collected the weights from multiple domain experts and used the averages. Hernández and Stolfo [3] provided users a high-level declarative language to specify arbitrarily complex decision rules, the so-called *equational theory*.

Rule-based approaches are powerful in capturing domain knowledge and are applicable to many cases where simple key equivalence cannot be found. However, specifying the rules requires deep understanding of the application domains and demands a time-consuming *knowledge acquisition* process and experimental evaluation. It is very difficult for human experts to provide a comprehensive set of decision rules, especially when *fuzzy comparisons* are involved.

Learning-based approaches, using statistical classification or machine learning techniques, automatically learn the decision rules from sample data. In these approaches, domain experts are required to provide classified examples rather than rules.

Newcombe et al. [12] and some others pioneered the statistical record linkage field. They proposed a probabilistic approach to discriminating matches and non-matches based on *odds ratios* of frequencies, which are computed based on intuition and past experience. Newcombe [13] summarized the development and application of their approach. Fellegi and Sunter [4] proposed a formal mathematical foundation for record linkage, which extends the *Bayes classification method* by maintaining acceptable error rates while leaving some cases unclassified. Newcombe et al.'s intuitive approach and Fellegi and Sunter's theory of record linkage have been the foundation of many software systems, including GRLS developed at Statistics Canada [14] and the record linkage system developed at the US Bureau of the Census [15]. Other statistical techniques, such as logistic regression [6], have also been used to learn entity identification rules.

Recently, decision tree techniques, such as CART [7] and C4.5 [1,2], have been applied in entity identification. Tejada et al. [8] further combined multiple decision trees via *bagging* to improve classification accuracy.

An obvious advantage of learning-base approaches over rule-based approaches is the elimination of the knowledge acquisition process. It is often much easier for users to provide manually classified examples than to specify rules. Many classification techniques have been developed in statistical pattern recognition, machine learning, and artificial neural networks and are potentially applicable to entity identification. There are also a variety of methods to combine multiple classification techniques to improve prediction accuracy. However, only a few of these methods have been applied in entity identification and the methods were chosen in an ad-hoc manner. In our approach, we select the best (base or composite) methods from a variety of widely used classification techniques and methods to combine multiple techniques via experiments.

3. Classification for entity identification

Given a pair of records from semantically corresponding tables in two databases, we need to determine whether they represent the same real-world entity. This is a *two-class* (match or non-match) *classification* problem. Each pair of records to be compared is described in terms of a vector of *features*, $\mathbf{x} = (x_1, x_2, \ldots, x_m)$. Each feature, x_i, is usually a distance or similarity measure for comparing the values of the two records on semantically corresponding attributes. The objective is to assign the record pair to one of two classes, match (M) or non-match (N), based on the features.

3.1. Classification techniques

A classification technique learns a general rule, called a *classifier*, from a set of sample record pairs, whose true classes are known. The learned classifier can then be used to predict the classes of other record pairs. The set of sample record pairs used to train a classifier is called a *training data set*. In practice, domain experts need to manually classify some record pairs, which are then used as training data. Various categories of classification methods, including statistical pattern recognition, machine learning, and artificial neural networks, have been theoretically analyzed and empirically evaluated [16]. Fig. 1 shows some widely used classification techniques.

The *Bayes method* estimates the odds ratio, $Pr(M|x)/Pr(N|x)$, and compares it to a threshold value to determine the class of a record pair. The threshold value is determined by the prior probabilities of the two classes and the relative costs associated with two types of errors, i.e., false matches and false non-matches. The Bayes method provides intuitively optimal classification. It, however, can seldom be applied directly without making simplifying assumptions because of the numerous combinations of feature values, especially when some of the features are continuous. *Naive Bayes*, which assumes that the features are conditionally independent, is commonly used. Continuous features usually need to be discretized prior to classification. When the conditional independence assumption about the features is seriously violated, second-order or even higher-order dependency terms may be considered to incorporate the dependencies among the features. Fellegi and Sunter's *record linkage theory* [4] extends the basic Bayes method to maintain acceptable levels of error rates. The

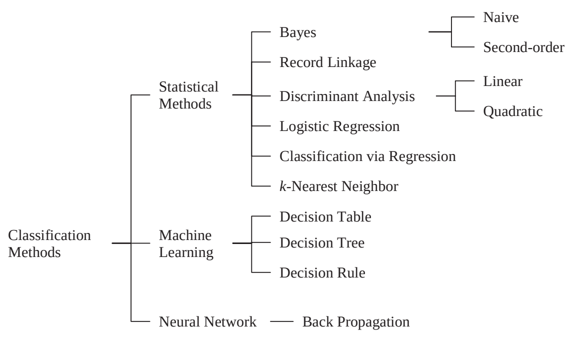

Fig. 1. Some classification techniques.

decision space is divided into three areas, match, non-match, and unclassified, based on two thresholds. Unclassified record pairs then need to be manually reviewed.

Fisher's *linear discriminant analysis* (LDA) is another widely used statistical classification method. LDA uses a line (in a two-dimensional feature space) or hyper-plane to separate the two classes. The coefficients in the linear model are chosen in such a way to separate the two classes as much as possible, assuming that the features follow normal distributions. LDA has been extended to produce more flexible decision boundaries. For example, *quadratic discriminant analysis* (QDA) uses a quadratic discriminant function to separate the two classes. Logistic regression assumes that the logarithm of the odds ratio of match to non-match is linearly related to the features; the decision boundary between the two classes is still linear in the original feature space. *Logistic regression* makes no assumption about the distributions of the features and has an advantage over LDA when many features are categorical. In practice, however, logistic regression and LDA often produce very similar results [16]. Regression techniques, such as *linear regression*, can also be used for classification. A regression model can be derived from the training data to predict the class using the features. *k-nearest neighbor* techniques simply memorize the training data and classify each new case into the majority class of the *k* cases in the training data that are closest to the new case.

Machine learning techniques generate *decision tables, trees,* or *rules* that are easily understood and are most compatible with human reasoning. A decision table learner selects several most discriminating features to form a lookup table, which is then used to classify new cases. Decision tree techniques follow a "divide and conquer" strategy and build tree-like sequential decision models. A decision tree can be easily translated into a set of mutually exclusive decision rules; each leaf node of the tree corresponds to a rule. There are also rule-induction techniques that can produce more general classification rules. A special form of simple classification rules is 1R (for "*1-rule*"), which uses a single most discriminating feature to determine the class of a case.

Back propagation is one of the most widely used neural network techniques for classification. Neural networks are highly interconnected networks, which learn by adjusting the weights of the connections between nodes on different layers. A neural network may have an *input layer*, an *output layer*, and zero or many *hidden layers*. Theoretically, a neural network with two or more hidden layers can approximate any function and has the potential to achieve the lowest possible error rates. However, neural networks are "black boxes"; it is hard to interpret the rules they follow in classifying cases. The training of a neural network is often not trivial; it takes experience and experimenting to adjust the parameters, such as the number of nodes on each layer and the learning rate.

3.2. Attribute-matching functions

Each pair of records to be compared is described in terms of a vector of features, each of which is usually a distance or similarity measure used to compare the values of the two records on semantically corresponding attributes. Given two (base or derived) attributes whose domains are A_1 and A_2, an *attribute-matching function* is a mapping $f : A_1 \times A_2 \rightarrow [0, 1]$, which returns the degree of match between two values drawn from A_1 and A_2, where 1 reflects a perfect match.

If two corresponding attributes in different databases share the same format and store accurate data, we can compare them using simple *equality*. However, we frequently observe both schema level and data level discrepancies. Semantically corresponding attributes often have different formats in different databases. There are incorrect data, phonetic errors, typographical errors, and different abbreviations in most operational databases. Human names are often misspelled or may be substituted with similar-sounding names. The same name may have different spellings in different languages, e.g., Joseph in English and Giuseppe in Italian, or different nicknames, e.g., Bob and Robert. Luján-Mora and Palomar [17] identified eleven types of data discrepancies across databases.

Transformation functions are needed to convert corresponding attributes into compatible formats.

Approximate attribute-matching functions are needed to measure the degree of similarity between two attribute values. There are many general-purpose approximate string-matching methods. The Soundex coding technique [18] has been widely used in record linkage problems to compute the phonetic distance between human names. Levenshtein's edit distance [19] is a simple yet widely used metric of string distance. There are also many special-purpose methods that are suitable for comparing special types of strings, e.g., human names and addresses [15]. Stephen [19] reviewed many string distance measures, including Hamming distance, Levenshtein's metrics, longest common substring, and q-grams. Budzinsky [18] compared 20 string comparison methods. Some of these methods account for spelling errors, such as insertion, deletion, transposition, and substitution of characters; some account for phonetic errors; some are special-purpose.

Numeric attributes measured on interval or ratio scales can be compared using normalized distance functions. Special dictionaries, or look-up tables, can be built to bridge different coding schemes used for semantically corresponding attributes in different databases. For example, name equivalence dictionaries can be built to account for different variants of human names, e.g., different nicknames, names in different languages, and different spellings of Asian names.

Different matching methods work well for different types of attributes. If multiple matching functions can be used to compare a pair of corresponding attributes, it may be necessary to evaluate various functions and select the most discriminating one for classification, because some classification techniques, e.g., linear discriminant analysis and logistic regression, are sensitive to highly correlated features. We can compute the correlation between each matching function and the class (match or non-match) of record pairs based on the training examples for classification and select the matching function that is most highly correlated with the class. There are also heuristics that help in choosing appropriate matching functions. For example, Soundex is good for comparing human names; edit distance measures are good for comparing strings with spelling errors; dictionaries are good to bridge different coding schemes. We can also construct arbitrarily complex matching functions by combining multiple matching functions and transformation functions.

4. A multiple classifier system approach to improving classification accuracy

A single classifier, regardless of how accurate it is, provides only one estimate of the optimal classification rule. Recently, there have been many efforts to combine multiple classifiers to obtain a better estimate of the optimal decision boundary and thus improve classification accuracy. Several conferences, such as Multiple Classifier Systems (MCS) have been devoted to such research [20–22].

We propose a multiple classifier system approach to improving classification accuracy in entity identification. Multiple classifiers are combined in a variety of ways, including *bagging*, *cross-validated committee*, *boosting*, *cascading*, and *stacking* (see Fig. 2), to determine whether a pair of records corresponds to the same entity in the real-world. These methods combine multiple classifiers of the same type (i.e., with *homogeneous base classifiers*) or different types (i.e., with *heterogeneous base classifiers*).

Methods for combining homogeneous classifiers gather the base classifiers into an *ensemble*, also called a *committee*, and ask the base classifiers to "vote" on the results [23]. The votes of base classifiers may or may not be weighted. In an un-weighted voting scheme, multiple classifiers trained independently using different training data sets are given equal weights in the voting. The data

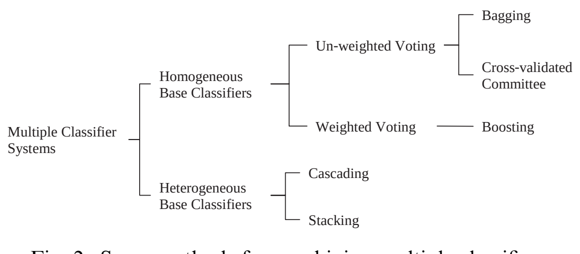
Multiple Classifier Systems
Homogeneous Base Classifiers
Heterogeneous Base Classifiers
Un-weighted Voting
Weighted Voting
Bagging
Cross-validated Committee
Boosting
Cascading
Stacking

Fig. 2. Some methods for combining multiple classifiers.

sets used to train the multiple base classifiers are generated based on an original training data set. In bagging (abbreviation for *bootstrap aggregating*), each training data set consists of *n* examples randomly sampled with replacement from the original training data set with *n* examples. The *n* examples in each generated training data set cover, on the average, 63.2% of the examples in the original training data set, with some examples appearing multiple times. This method for generating random training data sets is called *bootstrap*. In a cross-validated committee method, the original training data set is randomly divided into *m* disjoint subsets (folds); *m* training data sets are generated by leaving one fold out each time. 10-fold cross-validated committees are commonly used in practice.

Boosting is a widely used ensemble method, in which the votes of base classifiers are weighted according to their performances. In boosting, base classifiers are learned sequentially. Each new classifier is trained to perform better on examples classified incorrectly by previous classifiers. This is achieved by giving those examples higher weights. In the voting for the final classification decision, base classifiers are weighted according to their accuracy.

Heterogeneous base classifiers can be combined via cascading and stacking. First, the output of one classifier can be used as an artificial input feature together with the original features to train another classifier. Such a cascade of multiple classifiers may perform better than base classifiers learned independently [24]. A weakness of most rule-induction methods, such as C4.5 decision tree [25], is that they cannot learn intermediate concepts and abstractions from multiple variables [16]. A decision tree divides the feature space into regions, whose boundaries are always orthogonal to one of the dimensions. In a two-dimensional feature space, these regions are rectangles, whose sides are parallel to one of the axes; for a linearly separable data set, they must use numerous line segments to approximate the discriminant line. Discriminant functions learned by other discriminant analysis methods, such as Fisher's LDA and logistic regression, can be used as additional input features to train decision trees or rules, so that the decision regions have more flexible boundaries in the feature space.

Another method for combining heterogeneous base classifiers is stacking [26], which trains another classifier, called a meta classifier, to make the final classification decision based on the predictions of multiple base classifiers. Each training example for the meta classifier is described in terms of the outputs of the base classifiers and the true class.

A multiple classifier system is not restricted to containing only two levels. Several multiple classifier systems can be further combined. Complex multiple classifier systems with arbitrarily many levels can be developed if necessary. In practice, however, multiple classifier systems with more than three layers are seldom significantly more productive.

In this paper, we focus on classification accuracy and leave out the scalability of the subsequent application of learned classifiers in matching records in the entire heterogeneous data sources. Scalability is indeed a critical issue that must be addressed, especially when large data sources need to be matched. A straightforward pair-wise comparison procedure requires $N \times M$ comparisons when two data sources with N and M records, respectively, need to be matched, and is prohibitively time-consuming when N and M are not trivial. Past research has studied this issue. For example, the sorted-neighbor method proposed by Hernández and Stolfo [3] sorts or indexes the two tables on selected common (base or derived) attributes (e.g., first three characters of a person's last name), called a blocking factor, and compares only records in a limited sliding window with regard to the blocking factor (i.e., having similar values on the blocking factor). The number of comparisons is reduced to $(N + M) \times w$, where w is the size of the sliding window and is independent of N and M. The time complexity is reduced from $O(N \times M)$ to $O(N + M)$ (note that w is a constant). It may be necessary to apply the method multiple times with different blocking factors and combine the results of the multiple runs to reduce the risk of missing match record pairs due to potential errors on the blocking factors. This method can be applied with the classifiers learned using our proposed approach to match records in large heterogeneous data sources and is not further discussed in this paper. Interested readers may refer to Hernández and Stolfo [3].

5. Empirical evaluation

We have evaluated our multiple classifier system approach using several sets of real-world heterogeneous data sources. In this paper, we report on some of our experiments of the passenger matching procedure of an application service provider (ASP) for the airline industry. This ASP serves over 20 national and international airlines. It maintains a separate PNR (''passenger reservation'') database for each served airline. One particular service is the identification of potential duplicate passengers in PNRs of a single airline or across airlines. Many airlines, among many other companies, are also planning to build an integrated customer database by consolidating various heterogeneous data sources. The techniques we are evaluating are also useful in such customer relationship management (CRM) efforts.

Fig. 3 shows a conceptual model of an airline PNR database. The information relevant to passenger matching includes: passenger name, frequent flyer number, PNR confirmation information, address, phone, and itinerary segment. We used a snapshot of each of the databases for two airlines in our experiment. Table 1 shows the number of records in the tables of the two snapshorts.

5.1. Attribute-matching functions

The comparison of a pair of passengers was based on some relevant attributes. We used exact comparison for some attributes (e.g., city, state,

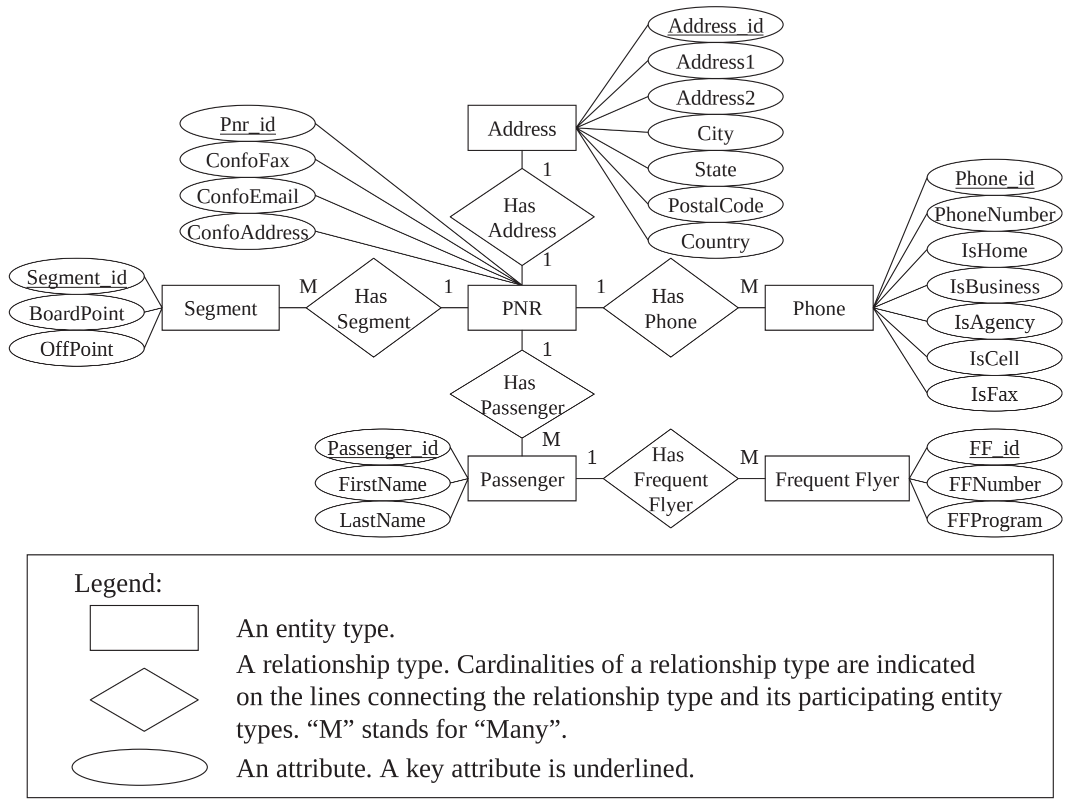

Fig. 3. A conceptual model of a PNR database.

Table 1
Number of records in the tables of the PNR database snapshots

Table name	Description	Number of records	
		Snapshot A	Snapshot B
PNR	Passenger reservation	261,141	545,763
Passenger	Passenger information	351,438	745,168
Frequent Flyer	Frequent flyer number	129,067	20,750
Address	Passenger address	117,797	235
Phone	Passenger phone numbers	600,149	756,033
Segment	Itinerary segments	953,930	1,294,707

boarding point, off point, confirmation email, confirmation fax, and phone numbers) and approximate comparison for some other attributes (e.g., address and confirmation address). There were various kinds of problems in passenger names (e.g., similar-sounding names, spelling errors, initials, variants of first name and middle name combinations, and nicknames). We combined multiple matching methods, including exact comparison, Soundex, sub-string, and edit distance, to compare passenger names. Postal codes had five or more digits. We compared two postal codes using exact comparison and sub-string. Table 2 summarizes these attribute-matching functions. For example, function Lname combined equality comparison, Soundex matching, substring matching, and Levenshtein's edit distance (see Section 2.2) to compare the passenger last names of two PNRs. The Pearson correlation coefficient of 0.950 indicates how well this attribute-matching function can predict whether two PNRs are about the same passenger. Function Lname was the most effective in such prediction; function Cell was the least effective.

5.2. Classification of record pairs

We trained classifiers to determine whether a pair of passengers matched or not, based on their distances on relevant attributes. Some passengers use frequent flyer numbers in their reservations. We relied on frequent flyer numbers to generate training examples. Two passengers with the same frequent flyer number for the same frequent flyer program are very likely to be the same person, while two passengers with different frequent flyer numbers for the same frequent flyer program are very likely to be different people. However, there are rare exceptions; different people, especially people in the same family, may share a single frequent flyer number, while a single person may use multiple frequent flyer numbers for a single program. We manually screened these exceptions from the training data set. The training data set consisted of 20,000 non-matching examples and 5000 matching examples.

We used some widely used classification techniques available in Weka [26], including 1-rule (1R), logistic regression (Logistic), classification via linear regression (Linear), J4.8 decision tree (J4.8), naive Bayes (Bayes), back propagation neural network (BP), and k-nearest neighbor (1-NN, 3-NN, and 5-NN), to classify record pairs. 1R selects the single most discriminating feature to make the classification decision. In this example, Lname was selected. The classification rules were:

IF Lname$\geqslant$0.61, Match; Otherwise, Non-Match.

The model learned by Logistic was:

IF $D_{\text{Logistic}} \geqslant 0$, Match; Otherwise, Non-Match,

where

$$\begin{aligned}D_{\text{Logistic}} = {} & 7.1489 \times \text{Fname} + 10.9680 \times \text{Lname} \\ & + 19.3729 \times \text{CFax} \\ & + 41994 \times \text{Cemal} \\ & + 3.1365 \times \text{Caddr} - 7.1046 \times \text{Street} \\ & + 3.8845 \times \text{City} + 0.8369 \times \text{State} \\ & + 13.7226 \times \text{Postal} \\ & + 1.3558 \times \text{Bpoint} + 1.0095 \times \text{Opoint} \\ & + 17.4604 \times \text{Home} + 14.9700 \times \text{Bus} \\ & + 2.3154 \times \text{Agency} + 12.8704 \times \text{Cell} \\ & + 10.8609 \times \text{Fax} - 15.0788.\end{aligned}$$

Table 2
Summary of attribute-matching functions

Name	Related attribute(s)	Description	Function	Range	r
Lname	Passenger.Last_Name	Passenger last name	Equality + Soundex + Substring + Edit Distance	[0, 1]	0.950
Fname	Passenger.First_Name	Passenger first name	Equality + Soundex + Substring + Edit Distance	[0, 1]	0.775
City	Address.City	City of address	Equality	{0, 1}	0.675
Caddr	PNR.ConfoAddress	Confirmation address	Edit Distance	[0, 1]	0.611
Street	Address.Address1 + Address.Address2	Street of address	Edit Distance	[0, 1]	0.607
State	Address.State	State of address	Equality	{0, 1}	0.580
Opoint	Segment.OffPoint	Off point of itinerary segment	Equality	{0, 1}	0.532
Cfax	PNR.ConfoFax	Confirmation fax	Equality	{0, 1}	0.497
Bpoint	Segment.BoardPoint	Boarding point of itinerary segment	Equality	{0, 1}	0.487
Postal	Address.PostalCode	Postal code of address	Equality + Substring	{0, 1}	0.362
Cemail	PNR.ConfoEmail	Confirmation email	Equality	{0, 1}	0.340
Bus	Phone.PhoneNumber + Phone.IsBusiness	Business phone number	Equality	{0, 1}	0.248
Agency	Phone.PhoneNumber + Phone.IsAgency	Agency phone number	Equality	{0, 1}	0.223
Home	Phone.PhoneNumber + Phone.IsHome	Home phone number	Equality	{0, 1}	0.215
Fax	Phone.PhoneNumber + Phone.IsFax	Fax number	Equality	{0, 1}	0.177
Cell	Phone.PhoneNumber + Phone.IsCell	Cell phone number	Equality	{0, 1}	0.129

r-Pearson correlation coefficient between an attribute-matching function and the class (match/non-match).
{0, 1}-The set consisting of two members 1 and 0.
[0, 1]-The closed real interval between 0 and 1.

The model learned by Linear was:

IF $D_{\text{Linear}} \geqslant 0$, Match; Otherwise, Non – Match,

where

$$\begin{aligned} D_{\text{Linear}} = {} & 0.1409 \times \text{Fname} + 0.7892 \times \text{Lname} \\ & + 0.0450 \times \text{CFax} + 0.0201 \times \text{Cemal} \\ & - 0.0755 \times \text{Caddr} - 0.0092 \times \text{Street} \\ & + 0.1665 \times \text{City} - 0.0070 \times \text{State} \\ & + 0.0360 \times \text{Postal} + 0.0186 \times \text{Bpoint} \\ & + 0.0261 \times \text{Opoint} + 0.0474 \times \text{Home} \\ & + 0.0337 \times \text{Bus} + 0.0157 \times \text{Agency} \\ & - 0.0391 \times \text{Fax} - 0.5939. \end{aligned}$$

Fig. 4 shows a decision tree generated by J4.8, Weka's implementation of C4.5 [25]. DecTab selected features Fname, Lname, State, and Opoint 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 (1-NN, 3-NN, and 5-NN) methods simply memorized the training examples and classified each new example according to the majority class of its k nearest training examples.

5.3. Comparison of techniques

We conducted experiments to compare the accuracy of different classification techniques and different combinations of techniques. We first ran each of ten base classification techniques 100 times; each time 66% of the 25,000 examples were randomly re-sampled for training; the rest were set aside for testing. We then tested whether various ways to combine multiple classifiers, including

```
Lname <= 0.6
|   Street <= 0.55: Non-Match (19737/13)
|   Street > 0.55
|       Cfax = -2: Match (2)
|       Cfax = -1: Non-Match (9)
|       Cfax = 0: Non-Match (14)
|       Cfax = 1: Match (3)
Lname > 0.6
    Fname <= 0.29
     |   Opoint = 0: Non-Match (164/1)
     |   Opoint = 1
     |       Fname <= 0.15: Non-Match (64/3)
     |       Fname > 0.15
     |           City = -2: Match (1)
     |           City = -1
     |           |   Cemail = -2: Non-Match (4/1)
     |           |   Cemail = -1: Match (7/1)
     |           |   Cemail = 0: Match (0)
     |           |   Cemail = 1: Match (0)
     |           City = 0: Non-Match (8/1)
     |           City = 1: Non-Match (0)
    Fname > 0.29: Match (4987/18)
```

Fig. 4. 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. An attribute-matching function returns: (−1) if one of two values under comparison is missing; and (−2) if both values are missing.

cascading, bagging, boosting, and stacking, could improve the performance of the base classifiers. Each composite classification method was also run 100 times; each time 66% of the 25,000 examples were randomly re-sampled for training, the rest were set aside for testing. Table 3 summarizes the accuracy and costs (i.e., training time and testing time) of the base and composite classifiers. For example, the mean and standard deviation of the accuracy of 1R were 98.921% and 0.085%, respectively; the mean and standard deviation of the false positive (i.e., PNRs about different passengers that are classified as matching) rate of 1R were 1.212% and 0.102%, respectively; the mean and standard deviation of the false negative (i.e., PNRs about the same passenger that are classified as non-matching) rate of 1R were 0.544% and 0.194%, respectively; the mean and standard deviation of the training time of 1R were 0.86 and 0.16 s, respectively; the mean and standard deviation of the testing time of 1R were 0.22 and 0.16 s, respectively.

Among the 26 base and composite classifiers, cascading Logistic and J4.8 was the most accurate (99.807%); 1R was the least accurate (98.921%). The testing time required by *k*-nearest neighbor methods, which do not learn any generalized structure from the training examples and compare every new example with every training example, is much longer than the testing time required by methods that learn some generalized structure. In this experiment, 1-NN, 3-NN, 5-NN spent 2319.51, 3197.25, and 3192.20 s on average in testing, respectively, while other methods spent between 0.22 (1R) and 2.41 (BP) s on average in testing. BP was much slower in training than any other method. The training time of BP was 515.42 s; the training times of other methods, except *k*-NN, were between 0.86 (1R) and 60.24 (DecTab) s.

We further compared the accuracy of the classifiers statistically using ANOVA and *t*-tests. An ANOVA test (Table 4) of the accuracy of the 26 classifiers indicates that at least two of the classifiers performed significantly differently in terms of accuracy ($F(25,2574) = 1809.657$, $p < 0.05$). A Sheffé's post hoc test (Table 5) recognized five homogeneous (in terms of accuracy) subsets of classifiers at the significance level $\alpha = 0.05$.

Table 6 summarizes a series of *t*-tests that compare the accuracy of each composite classifier with the accuracy of a base classifier. Bagging and boosting combine multiple classifiers of the same type into an ensemble, or committee, and take a vote among the members. It does not make much sense to bag or boost *k*-NN methods because they do not learn any generalized structure from training examples. We bagged nine classifiers of each of the other seven base classification techniques (i.e., 1R, Linear, J4.8, Logistic, DecTab, Bayes, and BP) and boosted a maximum of nine classifiers of each type using the AdaBoost.M1 method [26]. Bagging significantly improved J4.8 and never significantly degraded any base classifier. Boosting significantly improved 1R, Linear, and J4.8. However, unlike bagging, which never significantly degraded a base classifier, boosting did degrade some classifiers, including DecTab and Logistic.

While bagging and boosting combine multiple classifiers of the same type, cascading and stacking

Table 3
Summary of classification results

Method	N	Accuracy (%)		False positive rate (%)		False negative rate (%)		Training time (s)		Testing time (s)	
		Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
1R	100	98.921	0.085	1.212	0.102	0.544	0.194	0.86	0.16	0.22	0.16
Bayes	100	99.764	0.042	0.119	0.033	0.707	0.171	1.22	0.47	0.74	0.36
Linear	100	98.928	0.084	1.252	0.096	0.350	0.118	19.72	3.01	0.82	0.20
DecTab	100	99.784	0.050	0.095	0.038	0.702	0.217	60.24	11.72	0.60	0.19
Logistic	100	99.769	0.040	0.134	0.042	0.617	0.150	7.12	0.77	0.52	0.24
J4.8	100	99.781	0.046	0.120	0.043	0.613	0.195	4.08	0.48	0.24	0.18
BP	100	99.778	0.046	0.112	0.048	0.662	0.186	515.42	15.43	2.41	0.24
1-NN	100	99.749	0.147	0.179	0.162	0.539	0.382	0.17	0.06	2319.51	42.42
3-NN	100	99.773	0.038	0.150	0.051	0.534	0.148	0.15	0.04	3197.25	119.93
5-NN	100	99.785	0.044	0.133	0.046	0.545	0.179	0.15	0.04	3192.20	140.51
Cascading	100	99.807	0.045	0.099	0.042	0.570	0.195	4.90	0.76	0.26	0.40
Bag_1R	100	98.922	0.083	1.205	0.102	0.568	0.180	7.06	0.32	0.29	0.10
Bag_Bayes	100	99.756	0.046	0.126	0.037	0.716	0.174	8.46	2.89	3.79	1.21
Bag_Linear	100	98.928	0.084	1.252	0.096	0.350	0.118	169.55	13.15	5.06	0.25
Bag_DecTab	100	99.790	0.045	0.072	0.028	0.761	0.206	617.88	62.92	3.66	0.26
Bag_Logistic	100	99.769	0.041	0.138	0.044	0.607	0.148	73.64	4.50	2.50	0.38
Bag_J4.8	100	99.793	0.044	0.113	0.038	0.584	0.184	34.50	4.14	0.57	0.24
Bag_BP	100	99.782	0.041	0.106	0.039	0.667	0.176	4567.54	177.66	19.22	1.21
Boost_1R	100	99.677	0.068	0.202	0.066	0.804	0.253	10.04	0.61	0.21	0.06
Boost_Bayes	100	99.758	0.044	0.123	0.038	0.721	0.172	22.55	1.84	3.62	0.41
Boost_Linear	100	99.449	0.216	0.439	0.215	0.998	0.809	214.08	41.39	4.87	1.01
Boost_DecTab	100	99.734	0.075	0.138	0.092	0.779	0.325	1306.35	324.32	6.59	9.88
Boost_Logistic	100	99.759	0.044	0.150	0.053	0.609	0.153	51.12	16.06	1.44	0.49
Boost_J4.8	100	99.802	0.040	0.133	0.041	0.460	0.167	55.16	5.45	0.70	0.05
Boost_BP	100	99.778	0.046	0.105	0.040	0.688	0.180	1145.65	219.77	2.59	0.86
Stacking	100	99.791	0.042	0.117	0.043	0.578	1.620	6050.18	130.35	4.82	0.14

Table 4
ANOVA of the accuracy of 26 single or composite classifiers

	Sum of squares	df	Mean square	F	Sig.
Between groups	245.226	25	9.809	1809.657	0.000
Within groups	13.952	2574	0.005		
Total	259.178	2599			

are usually used to combine classifiers of different types. We used the discriminant function D_{Logistic} learned by Logistic as an additional input feature to train J4.8 again. The cascaded classifier performed significantly better than each of two base classifiers (compared with J4.8: $t(198) = 4.060$, $p < 0.05$; compared with Logistic: $t(198) = 6.335$, $p < 0.05$). Stacking treats the outputs of base classifiers as input features and trains a meta-learner, which can be a classifier of any type, to make the final classification decisions. We combined seven base classifiers, including 1R, Linear, J4.8, Logistic, DecTab, Bayes, and BP using stacking, and used logistic regression as the meta-learner. The stacked classifier performed better than every base classifier.

In another set of studies, we evaluated the electronic product catalogs (E-catalog) of two leading online bookstores and the property databases managed by two departments of a large public university. Cascading logistic regression and J4.8 decision tree performed better than the two base classifiers in both studies. Stacking seven classifiers of different types using logistic regression as the meta classifier was more accurate than the base classifiers in both studies. Bagging improved J4.8 and DecTab in the E-catalog example, and J4.8, DecTab, and Bayes in the property example. Bagging never degraded a base classification technique. Boosting improved 1R, Linear, J4.8, and DecTab but degraded Logistic and Bayes in the E-catalog example. Boosting improved only 1R but degraded Linear and J4.8 in the property example. Detailed results for these two cases are omitted in the paper to save space and are available from the authors.

Table 5
Sheffé's test of the accuracy of 26 classifiers—homogeneous subsets

Method	*N*	Subset for alpha = 0.05				
		1	2	3	4	5
1R	100	98.921				
Bag_1R	100	98.922				
Bag_Linear	100	98.928				
Linear	100	98.928				
Boost_Linear	100		99.449			
Boost_1R	100			99.677		
Boost_DecTab	100			99.734	99.734	
1-NN	100				99.749	99.749
Bag_Bayes	100				99.756	99.756
Boost_Bayes	100				99.758	99.758
Boost_Logistic	100				99.759	99.759
Bayes	100				99.764	99.764
Bag_Logistic	100				99.769	99.769
Logistic	100				99.769	99.769
3-NN	100				99.773	99.773
BP	100				99.778	99.778
Boost_BP	100				99.778	99.778
J4.8	100				99.781	99.781
Bag_BP	100				99.782	99.782
DecTab	100				99.784	99.784
5-NN	100				99.785	99.785
Bag_DecTab	100				99.790	99.790
Stacking	100				99.793	99.793
Bag_J4.8	100				99.795	99.795
Boost_J4.8	100					99.802
Cascading	100					99.807
Sig.		1.000	1.000	0.229	0.101	0.177

5.4. Summary of results

Past research in other problem domains has found that bagging often improves classification accuracy and seldom degrades it; boosting may perform significantly better than bagging, but may also significantly degrade performance [23,26]. Bagging works better for unstable classification techniques than for stable ones. Classifiers produced by unstable techniques (e.g., decision tree, decision table, decision rule, and neural network learners) may change significantly in response to small changes in the training data. Linear model learners, such as Fisher's LDA and logistic regression, are generally very stable. Boosting performs well in low-noise cases but overfits very badly in high-noise cases, while the performance of bagging is not affected by noise. One explanation is that each new classifier generated in boosting places a larger weight on previously incorrectly classified examples, which are very likely noises in high-noise cases.

Our experimental results agreed with these previous findings from other problem domains. Bagging improved some base classification techniques and never degraded a technique. Boosting sometimes performed significantly better than bagging but sometimes degraded some base

Table 6
t-tests that compare a composite classifier with a base classifier (df = 198)

Base method	Bagging to base		Boosting to base		Cascading to base		Stacking to base	
	t	*p*	*t*	*p*	*t*	*p*	*t*	*p*
1R	0.040	0.484	69.767	0.000			91.822	0.000
Linear	−0.020	0.492	22.512	0.000			91.630	0.000
J4.8	1.826	0.035	3.347	0.000	4.060	0.000	1.562	0.060
DecTab	0.929	0.177	−5.492	0.000			1.074	0.142
Logistic	−0.102	0.459	−1.802	0.037	6.335	0.000	3.731	0.000
Bayes	−1.161	0.124	−0.984	0.163			4.603	0.000
BP	0.666	0.253	0.000	0.500			2.068	0.020

classification techniques. Bagging worked better for unstable techniques such as decision tree, decision table, and naive Bayes than for stable ones such as linear model learners. Boosting was not productive in the high-noise (many training examples were wrong because of errors in a common key) property example except for the simplest technique, 1-R. In addition, our results show that cascading or stacking heterogeneous classification techniques always performed better than any individual base classifier.

We are aware of the limited generalizability of our empirical results. While we have evaluated three cases using real-world data in different domains and obtained encouraging results, more empirical studies, especially in less balanced and more difficult real-world situations, need to be conducted in the future to further validate the usefulness of the proposed approach in real-world applications.

6. Contributions and future work

We have described a novel multiple classifier system approach to entity identification for heterogeneous database integration. Our experimental results show that combining multiple classifiers in a variety of ways, such as cascading, bagging, boosting, and stacking, may improve classification accuracy, thus impacting database integration. Since the performance of various base and composite classification techniques varies in different situations, we recommend that practitioners conduct experiments to select the best techniques in each particular application. We have also generated some preliminary heuristics: bagging works better on unstable techniques than on stable techniques and seldom degrades performance; boosting may perform significantly better than bagging but may also degrades performance, especially in high-noise cases; cascading or stacking heterogeneous classifiers usually performs better than each base classifier.

Besides the methods for combining multiple classifiers we have described in this paper, it is also possible to combine automatically learned classifiers with manual classification rules. Decision rules learned by classification techniques and rules provided by human experts can be synthesized to take advantage of both the domain knowledge of the experts and the patterns revealed by the data.

In the experiments we have described in this paper, we compared different techniques in terms of plain accuracy using a somewhat balanced data set. In real-world applications, however, neither are the two classes (i.e., match and non-match) of record pairs balanced, nor are the costs associated with the two types of errors (i.e., false match and false non-match) symmetric. Different weights should be given to the two classes, according to the prior probabilities of the two classes and the relative costs of the two types of errors.

Partial classification methods for entity identification need to be further evaluated. In classical classification problems, classifiers are built to minimize error rates or costs of errors. If the error rates or costs of the classifiers are not acceptable, however, the classifiers cannot be trusted and have to be abandoned. The training effort is wasted. To avoid wasting the entire training effort, part of the classifiers may be used. Usually the classifiers tend to be less accurate near the boundaries between different classes. A different formulation of the classification problem is to classify as many new examples as possible while maintaining acceptable error rates. Examples that cannot be classified with the required degree of certainty can simply be rejected and further evaluation demanded. In other words, given acceptable error rates, the goal is to minimize the percentage of "unclassified" examples.

The techniques we have described in this paper are useful for detecting instance-level correspondences across data sources. A related problem is the identification of schema-level correspondences [27]. Techniques for solving the two problems can be incorporated into an iterative procedure, so that correspondences on the two levels can be evaluated incrementally [28].

References

[1] M. Ganesh, J. Srivastava, T. Richardson, Mining entity-identification rules for database integration, in: Proceedings of the KDD, 1996, pp. 291–294.

[2] V.S. Verykios, A.K. Elmagarmid, E.N. Houstis, Automating the approximate record-matching process, Inf. Sci. 126 (1–4) (2000) 83–98.

[3] M.A. Hernández, S.J. Stolfo, Real-world data is dirty: data cleansing and the merge/purge problem, Data Min. Knowledge Discovery 2 (1) (1998) 9–37.

[4] I.P. Fellegi, A.B. Sunter, Atheory of record linkage, JASA 64 (328) (1969) 1183–1210.

[5] W.E. Winkler, Matching and record linkage, in: Proceedings of the Record Linkage Techniques, 1997, pp. 374–403.

[6] J.C. Pinheiro, D.X. Sun, Methods for linking and mining massive heterogeneous databases, in: Proceedings of the KDD, 1998, pp. 309–313.

[7] I.J. Haimowitz, Ö. Gür-Ali, H. Schwarz, Integrating and mining distributed customer databases, in: Proceedings of the KDD, 1997, pp. 179–182.

[8] S. Tejada, C.A. Knoblock, S. Minton, Learning object identification rules for information integration, Inf. Syst. 26 (8) (2001) 607–633.

[9] A.L.P. Chen, P.S.M. Tsai, J.L. Koh, Identifying object isomerism in multidatabase systems, Distributed Parallel Databases 4 (2) (1996) 143–168.

[10] A. Segev, A. Chatterjee, A framework for object matching in federated databases and its implementation, International Journal of Cooperative Information Systems (IJCIS) 5 (1) (1996) 73–99.

[11] D. Dey, S. Sarkar, P. De, Entity matching in heterogeneous databases: a distance-based decision model, in: Proceedings of the HICSS, Hawaii, USA, 1998, pp. 305–313.

[12] H.B. Newcombe, J.M. Kennedy, S.J. Axford, A.P. James, Automatic linkage of vital records, Science 130 (3381) (1959) 954–959.

[13] H.B. Newcombe, Handbook of Record Linkage: Methods for Health and Statistical Studies, Administration, and Business, Oxford University Press, Oxford, 1988.

[14] M.E. Fair, Record linkage in an information age society, in: Proceedings of the Record Linkage Techniques, 1997, pp. 427–441.

[15] 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.

[16] S.M. Weiss, C.A. Kulikowski, Computer Systems That Learn—Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert System, Morgan Kaufmann, San Mateo, CA, USA, 1991.

[17] S. Luján-Mora, M. Palomar, Reducing inconsistency in integrating data from different sources, in: Proceedings of the IDEAS, 2001, pp. 209–218.

[18] C.D. Budzinsky, Automated spelling correction, Technical Report Statistics, Canada, 1991.

[19] G.A. Stephen, String Searching Algorithms. World Scientific, Singapore, 1994.

[20] J. Kittler, F. Roli (Eds.), Proceedings of the MCS 2000, Springer, Berlin, 2000.

[21] J. Kittler, F. Roli (Eds.), Proceedings of the MCS 2001, Springer, Berlin 2001.

[22] J. Kittler, F. Roli (Eds.), Proceedings of the MCS 2002, Springer, Berlin, 2002.

[23] T.G. Dietterich, Ensemble methods in machine learning, in: Proceedings of the MCS, Cagliari, Italy, 2000, pp. 1–15.

[24] J. Gama, P. Brazdil, Cascade Generalization, Mach. Learn. 41 (3) (2000) 315–343.

[25] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, USA, 1993.

[26] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation, Morgan Kaufmann, San Francisco, CA, USA, 2000.

[27] H. Zhao, S. Ram, Clustering database objects for semantic integration of heterogeneous databases, in: Proceedings of the AMCIS, Boston, MA, USA, August 2001, pp. 357–362.

[28] S. Ram, H. Zhao, Detecting both schema-level and instance-level correspondences for the integration of e-catalogs, in: Proceedings of the WITS, New Orleans, LA, USA, 2001, pp. 193–198.