Efficient Techniques for Online Record Linkage

Debabrata Dey, *Member*, *IEEE*, Vijay S. Mookerjee, and Dengpan Liu

Abstract—The need to consolidate the information contained in heterogeneous data sources has been widely documented in recent years. In order to accomplish this goal, an organization must resolve several types of heterogeneity problems, especially the *entity heterogeneity* problem that arises when the same real-world entity type is represented using different identifiers in different data sources. Statistical *record linkage* techniques could be used for resolving this problem. However, the use of such techniques for online record linkage could pose a tremendous communication bottleneck in a distributed environment (where entity heterogeneity problems are often encountered). In order to resolve this issue, we develop a matching tree, similar to a *decision tree*, and use it to propose techniques that reduce the communication overhead significantly, while providing matching decisions that are guaranteed to be the same as those obtained using the conventional linkage technique. These techniques have been implemented, and experiments with real-world and synthetic databases show significant reduction in communication overhead.

Index Terms—Record linkage, entity matching, sequential decision making, decision tree, data heterogeneity.

1 Introduction

The last few decades have witnessed a tremendous increase in the use of computerized databases for supporting a variety of business decisions. The data needed to support these decisions are often scattered in heterogeneous distributed databases. In such cases, it maybe necessary to link records in multiple databases so that one can consolidate and use the data pertaining to the same real-world entity. If the databases use the same set of design standards, this linking can easily be done using the primary key (or other common candidate keys). However, since these heterogeneous databases are usually designed and managed by different organizations (or different units within the same organization), there maybe no common candidate key for linking the records. Although it maybe possible to use common nonkey attributes (such as name, address, and date of birth) for this purpose, the result obtained using these attributes may not always be accurate. This is because nonkey attribute values may not match even when the records represent the same entity instance in reality.

The above problem—where a real-world entity type is represented by different identifiers in two databases—is quite common in the real world and is called the *entity heterogeneity* problem [13], [14] or the *common identifier* problem [8], [12]. The key question here is one of *record linkage*: given a record in a local database (often called the *enquiry record*), how do we find records from a remote database that may match the enquiry record? Traditional record linkage techniques, however, are designed to link an enquiry record with a set of records in a *local* master file. Given the enquiry record and a record from the (local) master file, these techniques compare the common nonkey attribute values of the two records to derive a similarity measure—typically the probability of a match [35] or the likelihood ratio [16]. If the similarity measure is above a certain threshold, the two records are said to satisfy the *linkage rule*.

Record linkage techniques have been widely used in real-world situations—such as health care [2], [17], [36], [40], immigration [4], [10], [11], and census [6], [20], [21], [37], [38]—where all the records are available locally. However, when the matching records reside at a remote site, existing techniques cannot be directly applied because they would involve transferring the entire remote relation, thereby incurring a huge communication overhead. As a result, record linkage techniques do not have an efficient implementation in an online, distributed environment and have mostly been confined to either local master files or to matching data from various sources in a batch processing mode. In order to fully appreciate the overall difficulty, two important characteristics of the problem context must be understood:

- *The databases exhibiting entity heterogeneity are distributed, and it is not possible to create and maintain a central data repository or warehouse where precomputed linkage results can be stored.* A centralized solution maybe impractical for several reasons. First, if the databases span several organizations, the ownership and cost allocation issues associated with the warehouse could be quite difficult to address. Second, even if the warehouse could be developed, it would be difficult to keep it up-to-date. As updates occur at the operational databases, the linkage results would become stale if they are not updated immediately. This staleness maybe unacceptable in many situations. For instance, in a criminal investigation, one maybe interested in the profile of crimes committed in the last 24 hours within a certain radius of the crime scene. In order to keep the warehouse current,

- *D. Dey is with Foster School of Business, University of Washington, Seattle, WA 98195-3200. E-mail: ddey@u.washington.edu.*
- *V.S. Mookerjee is with the University of Texas at Dallas, Richardson, TX 75803-0688. E-mail: vijaym@utdallas.edu.*
- *D. Liu is with the University of Alabama in Huntsville, Huntsville, AL 35899-0001. E-mail: Dengpan.Liu@uah.edu.*

Manuscript received 2 June 2009; accepted 23 Sept. 2009; published online 12 Aug. 2010.
Recommended for acceptance by B.C. Ooi.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-06-0480.
Digital Object Identifier no. 10.1109/TKDE.2010.134.

the sites must agree to transmit incremental changes to the data warehouse on a real-time basis. Even if such an agreement is reached, it would be difficult to monitor and enforce it. For example, a site would often have no incentive to report the insertion of a new record immediately. Therefore, these changes are likely to be reported to the warehouse at a later time, thereby increasing the staleness of the linkage tables and limiting their usefulness. In addition, the overall data management tasks could be prohibitively time-consuming, especially in situations where there are many databases, each with many records, undergoing real-time changes. This is because the warehouse must maintain a linkage table for each pair of sites, and must update them every time one of the associated databases changes.

- *The participating sites allow controlled sharing of portions of their databases using standard database queries, but they do not allow the processing of scripts, stored procedures, or other application programs from another organization.* The issue here is clearly not one of current technological abilities, but that of management and control. If the management of an organization wants to open its databases to outside scripts from other organizations, there are, of course, a variety of ways to actually implement it. However, the decision to allow only a limited set of database queries (and nothing more) is not based on technological limitations; rather it is often a management decision arising out of security concerns. More investment in technology or a more sophisticated scripting technique, therefore, is not likely to change this situation. A direct consequence of this fact is that the local site cannot simply send the lone enquiry record to the remote site and ask the remote site to perform the record linkage and send the results back.

In Section 2, we provide two real-world examples of the online record linkage problem. These examples clearly demonstrate that the two problem characteristics discussed above are quite common and are likely to be present in many other practical situations, as well.

Since the early work by Fellegi and Sunter [16], Newcombe and Kennedy [30], and Tepping [35], the issue of record linkage has received a lot of attention in statistics and computer science literature. However, most of the previous work has been in the context of record linkage accuracy: based on the structure of the data files, how can one make better (or more accurate) linkage decisions? For example, Jaro [21], in applying record linkage to census data, proposes the use of the EM algorithm for accurately estimating the probability parameters and applying them to obtain the matching probabilities that are later used in an assignment model. Copas and Hilton [10] consider different statistical models for observed records and fit them to previously matched immigration record pairs; once a model is fitted, it can be used to make automated linkage decisions for unmatched record pairs. Newcombe et al. [29] examine how partial matches in string attributes (such as the name of a person) can be used in probabilistic record linkage. The ability to use partial matches, as opposed to exact matches, mimics how a human expert performs matching and can greatly enhance the accuracy of linkage decisions. Belin and Rubin [6] question the common assumption of conditional independence among matching patterns of attributes and develop a method for calibrating the false-match rate in record linkage by fitting mixtures of transformed normal distributions. This mixturemodel calibration method is shown to work well when applied to test census data. Larsen and Rubin [23] use mixture models for accurate record linkage in an iterative manner. In their scheme, some record pairs are directly classified as "links" and "nonlinks" based on the available information, while other record pairs are sent for clerical review. As the clerical review corrects some of the erroneous classifications, this information is used for a reestimation of the mixture models. Winkler [39] extends the model of Fellegi and Sunter [16] with fewer assumptions and develops a linkage model that is computationally more efficient and requires less human intervention. Baxter et al. [5] show that two new blocking methods, which are used in record linkage systems to reduce the number of candidate record comparison pairs, can maintain or improve the record linkage accuracy. Minton et al. [26] describe a new machine learning approach that creates expert-like rules for field matching, a key subprocess in record linkage. In this approach, they define the relationship between two field values using a set of heterogeneous transformations, and can thus produce more accurate results by modeling more sophisticated relationships. In addition to the papers discussed above, there have been many others published in various journals and conferences; a recent summary is provided in [15] and [19]. While all these papers attempt to improve the accuracy of record linkage technique, to the best of our knowledge, there has not been any work that attempts to make the online record linkage process more efficient by reducing the communication overhead in a distributed environment. Therefore, the main contribution of our work is in the application of sequential information acquisition theory to develop rigorous techniques that can be used to perform record linkage in an efficient manner. To demonstrate that these techniques provide significant savings in communication overhead, we normalize the communication overhead needed by our approach by the size of the remote database from where the matching records need to be extracted. This provides a relative measure of performance that is more meaningful than merely reporting the absolute communication overhead needed by a technique to generate the required set of matching records.

An important issue associated with record linkage in distributed environments is that of schema integration. For record linkage techniques to work well, one should be able to identify the common nonkey attributes between two databases. If the databases are designed and maintained independently—as is the case in most heterogeneous environments—it would be necessary to develop an integrated schema before the common attributes can be identified. Schema integration is not the focus of this paper; however, there is an extensive body of literature that deals with this issue; interested readers are referred to [3], [7], [22], [24], [25], [31], [33], [34], among others. In this study, we assume that the attribute-level reconciliation has already been performed prior to employing the record linkage techniques discussed in this paper.

The rest of this paper is organized as follows: Section 2 describes two illustrative examples to motivate the problem studied in this paper. Section 3 develops the foundation for the proposed techniques, which are then analyzed in detail in Section 4. Results of detailed numerical experiments with real and synthetic data sets are reported in Section 5. Section 6 concludes the paper and offers possible directions for future research.

2 Motivational Examples

In order to motivate the problem context and illustrate the usefulness of the sequential approaches presented in this paper, we provide two real-world examples: the first one is drawn from insurance claims processing, and the second from crime investigation.

2.1 Example: Insurance Claims Processing

Consider the following situation in a large city with four major health insurance companies, each with several million subscribers. Each insurance company processes more than 10,000 claims a day; manual handling of this huge volume could take significant human effort resulting in high personnel and error costs. A few years ago, the health insurance companies and the medical providers in the area agreed to automate the entire process of claims filing, handling, payment, and notification. In the automated process, a medical service provider files health insurance claims electronically using information (about patients and services provided) stored in the provider database. A specialized computer program at the insurance company then processes each claim, issues payments to appropriate parties, and notifies the subscriber.

Although automated processing works well with most claims, it does not work with exceptions involving double coverage. A double coverage is defined as the situation where a person has primary coverage through his/her employer and secondary coverage through the employer of the spouse. Each service is paid according to a schedule of charges by the primary insurance; co-payments and nonallowable amounts are billed to the secondary insurance. An exceptional claim is one where the insurance company is billed as the primary for the first time, whereas previous billings to this company have been as the secondary. Quite often, medical service providers submit the primary claims incorrectly (to the secondary carrier). Therefore, when an exceptional claim is received, the insurance company would like to verify that it is indeed the primary carrier for the subscriber. Since the system cannot currently verify this, all exceptional claims are routed for manual processing.

The insurance companies request that their subscribers inform them of the existence of (and changes in) secondary coverage. However, many subscribers forget to send the appropriate notification to update the subscriber database. To complicate things further, different employers use different calendars for open enrollment—some use the calendar year, others use the fiscal year, and many academic institutions use the academic year. Furthermore, subscribers often change jobs, and their insurance coverages change accordingly. With rapid economic growth and the proliferation of double income families around the city, each insurance company receives several thousand updates per day to their subscriber database. However, since not all updates are propagated across the companies, stories of mishandled claims are quite common.

In order to overcome this problem, the insurance companies have recently agreed to partially share their subscriber databases with one another. Under this agreement, an insurance company would be able to see certain information (such as name, address, and employer) about all the subscribers in the other companies by using SQL queries. Of course, confidential information (such as social security number, existing diseases, and test results) would not be shared, nor would the processing of application programs or scripts from other companies be allowed. The companies have enhanced the existing claims processing software so that the subscriber information in all other databases can be consulted to determine the current state of coverage. Specifically, before processing an exceptional claim, one obtains the coverage information from the other companies, and checks for the existence of double coverage.

Unless an efficient technique is used, the communication burden needed for record linkage in the above environment maybe quite high. The databases are quite large—the average number of subscribers per company is more than a million. Each record contains many common attributes with a total size of about 500 bytes per record. If a more efficient technique is not used, even with a dedicated T-1 connection, it would take in excess of 40 minutes for downloading the common attribute values of all the records from a remote database (ignoring queuing delays and the framing overhead). Thus, the need for an efficient technique, such as the one we are about to propose here, is clearly indicated for this application.

2.2 Example: Crime Investigation

Consider the situation in a large metropolitan area consisting of about 40 municipal regions. Each municipality is equipped with (mostly incompatible) criminal data processing systems and their respective data models. Although, the municipalities share a significant portion of the stored criminal records among themselves, it has long been decided that it is not practical to create a central data warehouse that consolidates all the information. The justification for this decision was derived from the fundamental failure of the US Government's "Interstate Identification Index" (III) which is fed by the "State Criminal History Information System" (CHIS). An effort spanning more than a decade has consolidated only about 60 percent of the state criminal records. No status report beyond 2001 is available from the US Department of Justice, and whether the remaining 40 percent records will ever be consolidated is, therefore, an open question at present.

Currently, a police officer investigating a crime at the site makes a phone call to a backroom operator, who searches through the different databases to determine if certain offender types are known to be located in the call area of interest. The process is quite inefficient. First, it is often difficult for a police officer to relay the exact search requirements to the operator. Second, the police officer has to rely on the operator's expertise and intuition in modifying the search criteria based on the results of a previous query. Third, when the search criteria are satisfied by several records in several databases, relaying all the information back to the police officer over the phone is cumbersome, error-prone, and time-consuming. Finally, if all backroom operators are busy working on other investigations, an officer may have to wait for a long time before an operator becomes available to provide the necessary help.

In order to address this problem, a proposal is currently under consideration whereby the field personnel (such as investigating officers, certain social workers, and forensic experts) would be provided with handheld devices. The basic idea in this proposal is that a crime investigator should be able to quickly download relevant information (appropriate to the crime profile of the case at hand) on these devices, instead of having to rely on a backroom operator to do the necessary research.

Unfortunately, there are several challenges in implementing this proposal. First, since no centralized data warehouse exists, an investigating officer may have to send queries to several databases separately to download the relevant information. Second, the handheld devices do not have enough storage capacity to download all the remote databases in a batch process and store them locally. Third, the connection speed on these machines (based on a wireless networking infrastructure) is not very high, making it impossible to download millions of records on a real-time basis. Therefore, the practicality of the entire proposal depends on finding a way to download only the relevant criminal records to the handheld devices on demand.

3 Proposed Model

In this section, we draw upon the research in the area of sequential information acquisition [27], [28] to provide an efficient solution to the online, distributed record linkage problem. The main benefit of the sequential approach is that, unlike the traditional *full-information* case, not all the attributes of all the remote records are brought to the local site; instead, attributes are brought one at a time. After acquiring an attribute, the matching probability is revised based on the realization of that attribute, and a decision is made whether or not to acquire more attributes. By recursively acquiring attributes and stopping only when the matching probability cannot be revised sufficiently (to effect a change in the linkage decision), the sequential approach identifies, as possible matches, the same set of records as the traditional full-information case (where all the attributes of all the remote records are downloaded). Before we discuss the sequential approach in more detail, some basic notation and an overview of traditional record linkage is necessary.

3.1 Basic Notation

Let a be an enquiry record at the local site, and let $R = \{b_1, b_2, \dots, b_n\}$ be a set of records at the remote site. We are interested in identifying the records in R that are possible matches of a. We consider a set of attributes $\mathcal{Y} = \{Y_1, Y_2, \dots, Y_K\}$ common to both a and R. The Y_k-value of a record r is denoted by $r(Y_k)$. The comparison results between two records, a and $b \in R$, and their common attributes can be expressed as the following random variables:

$$M = \begin{cases} 1, & \text{if } a \text{ and } b \text{ are linked,} \\ 0, & \text{otherwise,} \end{cases}$$
$$\text{and} \quad U_k = \begin{cases} 1, & \text{if } a(Y_k) = b(Y_k), \\ 0, & \text{otherwise,} \end{cases} \quad k \in \{1, 2, \dots, K\}.$$

Although U_k is represented as a binary-valued random variable here, it is straightforward to extend this idea to the case where U_k can assume more than two values. In that case, we would be able to express partial matches between attribute values as well.

The possible match between a and b is quantified by the conditional probability that the two records refer to the same real-world entity instance, given $\mathbf{U} = (U_1, U_2, \dots, U_K)$, the matching pattern of their recorded attribute values; this probability can be estimated using Bayes' conditionalization formula:

$$\begin{aligned} p(\mathbf{U}) &= \Pr[M = 1|\mathbf{U}] \\ &= \frac{\Pr[\mathbf{U}|M = 1]\Pr[M = 1]}{\Pr[\mathbf{U}|M = 1]\Pr[M = 1] + \Pr[\mathbf{U}|M = 0]\Pr[M = 0]} \\ &= \left(1 + \frac{1 - p|_\emptyset}{p|_\emptyset}\frac{1}{L(\mathbf{U})}\right)^{-1}, \end{aligned} \tag{1}$$

where $L(\mathbf{U}) = \frac{\Pr[\mathbf{U}|M=1]}{\Pr[\mathbf{U}|M=0]}$ is the likelihood ratio for the matching pattern $\mathbf{U}$, and $p|_\emptyset = \Pr[M = 1]$ denotes the prior probability that a and b refer to the same real-world entity; clearly, $\Pr[M = 0] = 1 - p|_\emptyset$.

In practice, it is quite common to make the simplifying (Naïve Bayes) assumption of conditional independence among U_ks given M [16], [21]. Equation (1) then simplifies to

$$p(\mathbf{U}) = \left(1 + \frac{1 - p|_\emptyset}{p|_\emptyset}\prod_{k=1}^{K}\frac{\Pr[U_k|M = 0]}{\Pr[U_k|M = 1]}\right)^{-1}. \tag{2}$$

Therefore, the parameters required to calculate p are: $p|_\emptyset$, $\Pr[U_k|M = 1]$, and $\Pr[U_k|M = 0]$, $k = 1, 2, \dots, K$. These parameters can be easily estimated and stored based on a matched set of training data [16]. Given any two records to be matched, the value of the matching probability p can be calculated based upon the values observed for U_k, $k = 1, 2, \dots, K$.

Traditionally, the linkage rule is expressed in terms of the likelihood ratio $L(\mathbf{U})$: any two records with the matching pattern $\mathbf{U}$ are not linked if $L(\mathbf{U}) < \theta$, and are linked as possible matches (perhaps requiring further clerical review) if $L(\mathbf{U}) \geq \theta$, where θ is a constant determined in order to minimize the total number of errors made in the linkage decision [16]. We make two important observations in this regard. First, we note that the condition $L(\mathbf{U}) \geq \theta$ is equivalent to the probability condition: $p(\mathbf{U}) \geq \alpha$, when $\theta = \frac{\alpha(1-p|_\emptyset)}{p|_\emptyset(1-\alpha)}$. Second, we express the threshold α (and hence θ) as a parameter of an explicit cost-benefit trade-off. In order to do that, consider the case of evaluating a possible linkage between two records a and b having a matching pattern $\mathbf{U}$. If a is the same as b ($a \simeq b$) and the

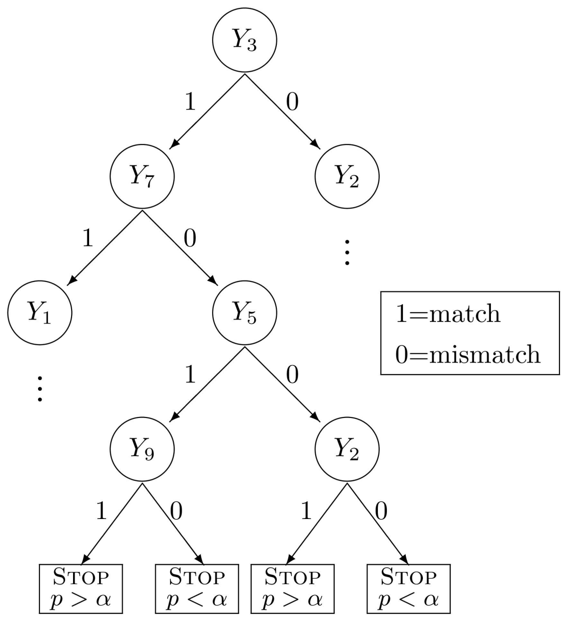


Fig. 1. A sample tree showing attribute acquisition order.

records are linked, or if $a \not\simeq b$ and the records are not linked, then there is no error. However, if $a \simeq b$, and we fail to link the records, a type-I error (false negative) is committed; let c_1 denote the cost of this error. Similarly, a type-II error (false positive) occurs when $a \not\simeq b$, but the records are linked; let c_2 denote the associated cost. A rational choice would be to link a and b if the total expected cost of linking them is lower than that of not linking them: $(1 - p(\mathbf{U}))c_2 \leq p(\mathbf{U})c_1$. Simplifying, we get the revised linkage rule:

$$p(\mathbf{U}) \geq \alpha = \frac{c_2}{c_1 + c_2}, \tag{3}$$

where $\alpha \in [0, 1]$ is the relative cost of type-II error. It is possible that a set of multiple remote records satisfy the linkage rule in (3). When this happens, the eventual matching could be decided from this set, perhaps after a clerical review.

3.2 Sequential Record Linkage and Matching Tree

The sequential approach decides on the next "best" attribute to acquire, based upon the comparison results of the previously acquired attributes. The acquisition of attributes can be expressed in the form of a *matching* tree as shown in Fig. 1. This tree can be used in the following manner: Starting at the root, we acquire attribute Y_3 first. If there is a match on this attribute, we acquire attribute Y_7; otherwise, we acquire Y_2. Similarly, after acquiring Y_7, if there is a match, we acquire Y_1, and so on, till a "STOP" node is reached. In the end, we would have a set of probability numbers for each remote record, based only on a subset of attributes that would have been acquired along a path of the tree. We now discuss how one can induce a matching tree similar to the one shown in Fig. 1.

There are two basic principles used in the induction of a matching tree: 1) input selection and 2) stopping. Before we describe these two principles, we would like to clarify an important point. In inducing the tree, as well as in our subsequent numerical analysis, we make the common assumption of conditional independence among U_ks given M; this reduces the overall computational burden. However, the idea presented here is more general, because, even in situations where this assumption does not hold, the matching tree can still be constructed through recursive partitioning of the training data, as is done in the traditional induction of a decision tree [32].

3.2.1 *Input Selection*

Assume that we are at some node of the tree and are trying to decide how to branch from there. Let $\mathcal{V}$ be the set of attributes that has already been acquired; the possibility that $\mathcal{V} = \emptyset$ is not excluded. The matching probability as revised by the attributes in $\mathcal{V}$ can be written as

$$p|_{\mathcal{V}} = \left(1 + \frac{1 - p|_{\emptyset}}{p|_{\emptyset}} \prod_{Y_j \in \mathcal{V}} \frac{\Pr[U_j|M = 0]}{\Pr[U_j|M = 1]}\right)^{-1}. \tag{4}$$

At this point, we would be interested in finding the next best attribute Y_k to be acquired from the set of remaining attributes, $\mathcal{Y} - \mathcal{V}$. There are two possibilities for the Y_k-value of a remote record $b \in R$: either $b(Y_k) = a(Y_k)$ or $b(Y_k) \neq a(Y_k)$, i.e., U_k is either one or zero. Of course, once we know Y_k, and hence U_k, we could revise the matching probability to

$$p|_{\mathcal{V} \cup \{Y_k\}} = \left(1 + \frac{1 - p|_{\mathcal{V}}}{p|_{\mathcal{V}}} \frac{\Pr[U_k|M = 0]}{\Pr[U_k|M = 1]}\right)^{-1}. \tag{5}$$

The revised probability value $p|_{\mathcal{V}\cup\{Y_k\}}$ could be higher or lower than $p|_{\mathcal{V}}$ depending on the actual realization of U_k. The probability of U_k having a certain value can be found conditioned on the matching pattern of the attribute values that have already been acquired and compared. Let $\mathcal{U} = \{U_j | Y_j \in \mathcal{V}\}$. Then, this probability can be written as

$$\begin{aligned}\Pr[U_k|\mathcal{U}] &= \sum_{m \in \{0,1\}} \Pr[U_k|M = m, \mathcal{U}] \Pr[M = m|\mathcal{U}] \\ &= \sum_{m \in \{0,1\}} \Pr[U_k|M = m] \Pr[M = m|\mathcal{U}] \\ &= \Pr[U_k|M = 1] p|_{\mathcal{V}} + \Pr[U_k|M = 0](1 - p|_{\mathcal{V}}).\end{aligned} \tag{6}$$

We now discuss the reduction in error cost as a result of acquiring Y_k. First, we simplify the notation. Let the current value of the matching probability $p|_{\mathcal{V}}$ be π; it is easily obtained from (4). Now, if Y_k is acquired at this point, the revised matching probability $p|_{\mathcal{V} \cup \{Y_k\}}$ can be obtained from (5). This revised probability is likely to be different from the current probability π: If the actual realization of U_k is favorable toward matching, then the matching probability would increase. If, on the other hand, the actual realization of U_k is not favorable, then the matching probability would decrease. In other words, $p|_{\mathcal{V}\cup\{Y_k\}}$ will either be $\pi' \geq \pi$ (with a probability of γ) or $\pi'' \leq \pi$ (with a probability of $1 - \gamma$). Of course, the value of the branching probability, γ, corresponds to the actual realization of U_k; it can be computed from (6) and must satisfy: $\pi = \gamma\pi' + (1 - \gamma)\pi''$. For example, if $U_k = 1$ is the favorable realization, then $\gamma = \Pr[U_k = 1|\mathcal{U}]$ and $1 - \gamma = \Pr[U_k = 0|\mathcal{U}]$, both of which can be directly obtained from (6). Based on this, the reduction in the error cost from acquiring Y_k is given by

$$\begin{aligned}\Delta\text{Cost} = {} & \text{Cost for } \pi - [\gamma \times \text{Cost for } \pi'] \\ & - [(1-\gamma) \times \text{Cost for } \pi''],\end{aligned}$$

where the cost associated with a matching probability of π can be written as

$$\text{Cost for } \pi = \begin{cases} \alpha(1-\pi), & \text{if } \pi > \alpha, \\ \pi(1-\alpha), & \text{otherwise.} \end{cases}$$

Analogous expression can be written for π' and π'' as well, with appropriate substitutions. Four possible cases arise:

- **Case 1** ($\pi > \alpha$ **and** $\pi'' > \alpha$): Of course, $\pi' \geq \pi > \alpha$. Therefore,
$$\begin{aligned}\Delta\text{Cost} &= \alpha(1-\pi) - \gamma\alpha(1-\pi') - (1-\gamma)\alpha(1-\pi'') \\ &= 0.\end{aligned}$$
- **Case 2** ($\pi > \alpha$ **and** $\pi'' \leq \alpha$): Again, $\pi' \geq \pi > \alpha$. Therefore,
$$\begin{aligned}\Delta\text{Cost} &= \alpha(1-\pi) - \gamma\alpha(1-\pi') - (1-\gamma)\pi''(1-\alpha) \\ &= (1-\gamma)(\alpha - \pi'') \geq 0.\end{aligned}$$
- **Case 3** ($\pi \leq \alpha$ **and** $\pi' \leq \alpha$): Of course, $\pi'' \leq \pi \leq \alpha$. Therefore,
$$\begin{aligned}\Delta\text{Cost} &= \pi(1-\alpha) - \gamma\pi'(1-\alpha) - (1-\gamma)\pi''(1-\alpha) \\ &= 0.\end{aligned}$$
- **Case 4** ($\pi \leq \alpha$ **and** $\pi' > \alpha$): Again, $\pi'' \leq \pi \leq \alpha$. Therefore,
$$\begin{aligned}\Delta\text{Cost} &= \pi(1-\alpha) - \gamma\alpha(1-\pi') - (1-\gamma)\pi''(1-\alpha) \\ &= \gamma(\pi' - \alpha) \geq 0.\end{aligned}$$

Therefore, for each attribute that has not been acquired, we can calculate the savings in error cost (ΔCost) if that attribute is acquired. We then pick the attribute with the highest savings. A special situation, however, arises when the cost savings are the same for all remaining attributes. When this happens, we use entropy reduction (instead of cost savings) as the selection criterion; we calculate the entropy reduction as [1]:

$$\begin{aligned}\Delta\text{Entropy} = {} & -[\pi \log \pi + (1-\pi)\log(1-\pi)] \\ & + \gamma[\pi' \log \pi' + (1-\pi')\log(1-\pi')] \\ & + (1-\gamma)[\pi'' \log \pi'' + (1-\pi'')\log(1-\pi'')],\end{aligned}$$

and pick the attribute that provides the largest entropy reduction (ΔEntropy).

3.2.2 Stopping

Now we consider the issue of when to stop expanding the matching tree. The stopping decision is made when no realization of the remaining attributes can sufficiently revise the current matching probability so that the matching decision changes. To that end, we find the upper and lower bounds of the eventual matching probability.

The calculation of these bounds is straightforward. Let $\mathcal{V}$ be the set of attributes already acquired, i.e., the current matching probability is $p|_{\mathcal{V}}$. Now, consider acquiring an attribute $Y_j \in (\mathcal{Y} - \mathcal{V})$. There are two possibilities: U_j is either one or zero. Of these two possible realizations of U_j, as mentioned earlier, one enhances the matching probability and is called a *favorable* realization, denoted u_j^+; the other realization, denoted u_j^-, is called an *unfavorable* realization. The favorable and unfavorable realizations of an attribute can be found from the training data—these realizations must satisfy

$$\frac{\Pr[Uj = u_j^+ | M = 1]}{\Pr[Uj = u_j^+ | M = 0]} \geq 1, \quad \text{and} \quad \frac{\Pr[Uj = u_j^- | M = 1]}{\Pr[Uj = u_j^- | M = 0]} < 1.$$

Based on these realizations, we can find the upper and lower bounds, denoted by $p^U|_{\mathcal{V}}$ and $p^L|_{\mathcal{V}}$, respectively, if $p|_{\mathcal{V}}$ is revised using all the remaining attributes:

$$\begin{aligned} p^U|_{\mathcal{V}} &= \left(1 + \frac{1 - p|_{\mathcal{V}}}{p|_{\mathcal{V}}} \prod_{Y_j \in (\mathcal{Y} - \mathcal{V})} \frac{\Pr[U_j = u_j^+ | M = 0]}{\Pr[U_j = u_j^+ | M = 1]}\right)^{-1}, \quad \text{and} \\ p^L|_{\mathcal{V}} &= \left(1 + \frac{1 - p|_{\mathcal{V}}}{p|_{\mathcal{V}}} \prod_{Y_j \in (\mathcal{Y} - \mathcal{V})} \frac{\Pr[U_j = u_j^- | M = 0]}{\Pr[U_j = u_j^- | M = 1]}\right)^{-1}. \end{aligned}$$

Note that $p^L|_{\mathcal{V}} \leq p|_{\mathcal{V}} \leq p^U|_{\mathcal{V}}$. Therefore, if $p^U|_{\mathcal{V}} < \alpha$, then it would be impossible to obtain a matching probability above α and change the current matching decision ($M = 0$). Similarly, if $p^L|_{\mathcal{V}} > \alpha$, then the matching probability would always be above α, so the current matching decision ($M = 1$) cannot be revised. In either case, the matching decision cannot be changed by acquiring more attributes; hence, we should stop expanding the tree. To put it in another way, we continue expanding the tree as long as there is any chance that the matching probability can be revised sufficiently to change the matching decision. This ensures that we do not stop expanding the tree prematurely. Any remote record, when matched using this tree, can follow exactly one path resulting in either $M = 1$ or $M = 0$, but not both. Because our stopping condition guarantees that the matching decision (M) would never change even if more attribute values are considered, the tree can be used to partition a set of remote records unambiguously; we call this the *completeness* property of the matching tree.

A distinct advantage of the tree-based sequential record linkage is that the matching tree can be precomputed and stored, thereby saving computational overhead at the time of answering a linkage query. Of course, in a real-world situation, one may need to store several matching trees, each for a different value of α. This is because the relative cost of a type-II error maybe different for different tasks within the same application. By identifying these tasks a priori and storing the matching trees for the appropriate values of α, one can avoid the extra computation at query time.

4 Tree-Based Linkage Techniques

In this section, we develop efficient online record linkage techniques based on the matching tree induced in Section 3.2. The overall linkage process is summarized in Fig. 2. The first two stages in this process are performed offline, using the

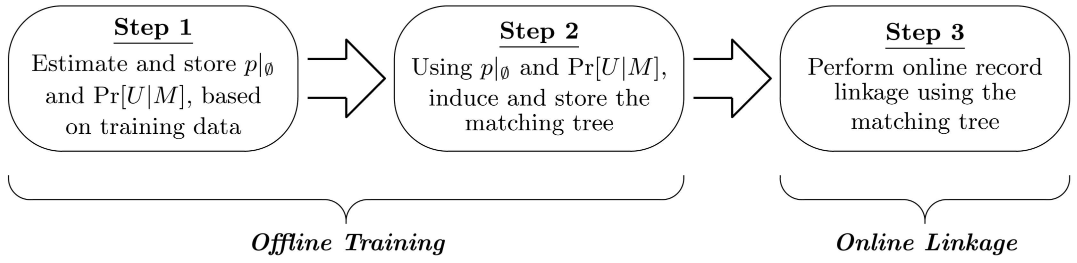

Step 1
Estimate and store $p|_\emptyset$ and $\Pr[U|M]$, based on training data
Step 2
Using $p|_\emptyset$ and $\Pr[U|M]$, induce and store the matching tree
Step 3
Perform online record linkage using the matching tree
Offline Training
Online Linkage

Fig. 2. The overall process of online tree-based linkage.

training data. Once the matching tree has been built, the online linkage is done as the final step.

We can now characterize the different techniques that can be employed in the last step. Recall that, given a local enquiry record, the ultimate goal of any linkage technique is to identify and fetch all the records from the remote site that have a matching probability of α or more. In other words, one needs to partition the set of remote records into two subsets: 1) *relevant* records that have a matching probability of α or more, and 2) *irrelevant* records that have a matching probability of less than α. Our aim is to develop techniques that would achieve this objective while keeping the communication overhead as low as possible. The partitioning itself can be done in one of two possible ways: 1) sequential, or 2) concurrent (see Fig. 3).

In sequential partitioning, the set of remote records is partitioned recursively, till we obtain the desired partition of all the relevant records. This recursive partitioning can be done in one of two ways: 1) by transferring the attributes of the remote records and comparing them locally, or 2) by sending a local attribute value, comparing it with the values of the remote records, and then transferring the identifiers of those remote records that match on the attribute value. As shown in Fig. 3, we call the first one *sequential attribute acquisition*, and the second, *sequential identifier acquisition*.

In the concurrent partitioning scheme, the tree is used to formulate a database query that selects the relevant remote records directly, in one single step. Hence, there is no need for identifier transfer. Once the relevant records are identified, all their attribute values are transferred. We call this scheme *concurrent attribute acquisition* (see Fig. 3).

4.1 Sequential Attribute Acquisition (SAA)

As mentioned earlier, in this technique, we acquire attribute from the remote records in a sequential fashion. Consider the

	Partitioning Scheme	
Transferred	*Sequential*	*Concurrent*
Attribute	Sequential Attribute Acquisition	Concurrent Attribute Acquisition
Identifier	Sequential Identifier Acquisition	Not Applicable

Fig. 3. Possible tree-based linkage techniques.

matching tree in Fig. 1. Working with this tree, we would first acquire attribute Y_3 for all the remote records in R. When the Y_3-value of $b \in R$ is compared to that of the local enquiry record a, we would get either a match or a mismatch. Therefore, the set of remote records in R gets partitioned into two sets R_1 and R_2, where R_1 is the set of records with matching Y_3-value, and $R_2 = R - R_1$. For all the records in R_1, the next attribute to acquire should be Y_7, whereas Y_2 should be acquired for all the records in R_2. Now, we would partition R_1 and R_2 based on the matching pattern of Y_7 and Y_2, respectively, and continue in this fashion.

Although the above procedure is simple, its implementation requires one to maintain the indices of the partitioned sets, say R_1 and R_2. For example, when querying the remote database for the Y_7-values of the records in R_1, we would need to identify which records belong to the set R_1. Since R is partitioned locally, a record identification scheme common to both the sites must be established. This can be easily done by using a candidate key from the remote database. In this scheme, during the first transfer, we acquire the identifiers for all the remote records and use these identifiers to specify any desired partition of the set of remote records.

The total communication overhead of the SAA technique is composed of three elements: 1) the transfer of attribute values from the remote to the local site, 2) the transfer of all the identifiers between the remote and the local sites, and 3) the transfer of those records that have a matching probability greater than α. It is possible to estimate the expected size of each of these three overheads from the matching tree. Let X be the set of nonleaf nodes in the matching tree, and let $q(x)$ be the probability of visiting node $x \in X$. Since there are n remote records, the expected number of remote records visiting node x should be $nq(x)$. Therefore, total attribute overhead should be $n \sum_{x\in X} q(x)s(x)$, where $s(x)$ is the size of the attribute acquired at node x.

In order to estimate the identifier overhead, we know that, at node x, $nq(x)$ identifiers must be sent from the local to the remote site, except at the root node where all n identifiers are received from (and not sent to) the remote site. Therefore, the expected number of identifiers sent to the remote site is $n \sum_{x\in X} q(x) - n$, whereas n identifiers are received. Assuming that the identifier size is s_K, the total identifier overhead should be $ns_K \sum_{x\in X} q(x)$.

Finally, given a remote record, the probability that it matches the local enquiry record is $p|_\emptyset$. So the expected number of remote records that would match the local record is $np|_\emptyset$. Denoting the size of each remote record by s_R, we get the total overhead from included records as $np|_\emptyset s_R$.

To provide a relative measure of the communication overhead, we normalize the required overhead by the size of the remote database, ns_R. Therefore, we can express the total overhead as a fraction of this size as

$$\text{Normalized Total Overhead} = \frac{\sum_{x\in X} q(x)s(x) + s_K \sum_{x\in X} q(x)}{s_R} + p|_\emptyset. \quad (7)$$

The terms $\sum_{x\in X} q(x)s(x)$ and $\sum_{x\in X} q(x)$ in (7) can be easily estimated from the matching tree induced using the training data.

4.2 Sequential Identifier Acquisition (SIA)

Sequential identifier acquisition is a minor variation of sequential attribute acquisition, but can provide significant savings in terms of the communication overhead. Its better performance lies in the fact that nonkey attributes stored in a database are often much larger than an identifier (typically 2 or 4 bytes long). If the attribute transfer could be replaced by identifier transfer, the total communication maybe reduced. Therefore, in this approach, we do not transfer the attribute values from the remote site. Rather, we send the local attribute value in a database query and ask the remote site to send the identifiers of all records that match the local attribute value. In order to see how this can be done, consider the matching tree in Fig. 1. Using this scheme, we would first ask for the identifiers of all the remote records in R. We would then send the Y_3-value of the local enquiry record and ask for the identifiers of those records that match on Y_3. The set of records identified by these identifiers is R_1; and R_2 is calculated as $R - R_1$. Then, we would send the identifiers in R_1 along with the Y_7-value of the local enquiry record, and ask the remote database to send the identifiers from only that subset of R_1 which matches on Y_7. Proceeding this way, we can eventually find the identifiers of all the remote records with a matching probability greater than α.

In this case also, there are three types of communication overheads: 1) attribute overhead, 2) identifier overhead, and 3) included record overhead. In order to obtain the total attribute overhead in this case, we note that the attribute value of the enquiry record at a node x must be sent as long as there is even a single remote record that visits x. If we assume that the event of a remote record visiting a node x is independent of another remote record visiting x, then the probability that at least one record will visit x is $1-(1-q(x))^n$. Therefore, the total attribute overhead should be $\sum_{x\in X}[1-(1-q(x))^n]s(x)$.

In order to find the total identifier overhead, note that the expected number of remote records visiting a node x is $nq(x)$. The identifier value of all these records must be transmitted from the local to the remote site along with the attribute value; this results in a total transfer of $nq(x)s_K$. In response, the remote site sends the identifiers of only those records that visit $l(x)$, the left child of x; the size of that transfer is $nq(l(x))s_K$. The only exception is the root node. At the root node, we receive the identifiers of all records, and do not have to resend these identifiers. The total can, therefore, be calculated as

$$\begin{aligned}
&\text{Identifier Overhead}\\
&\quad = \text{Identifier Sent} + \text{Identifier Received}\\
&\quad = \left[n\sum_{x\in X} q(x)s_K - ns_K\right] + \left[n\sum_{x\in X} q(l(x))s_K + ns_K\right]\\
&\quad = ns_K \sum_{x\in X}[q(x) + q(l(x))].
\end{aligned}$$

The included record overhead is still given by $np|_\emptyset s_R$. Normalizing the total overhead by the size of the remote database, we have

$$\begin{aligned}
&\text{Normalized Total Overhead}\\
&\quad = \frac{\frac{1}{n}\sum_{x\in X}[1-(1-q(x))^n]s(x) + s_K \sum_{x\in X}[q(x)+q(l(x))]}{s_R}\\
&\qquad + p|_\emptyset.
\end{aligned} \quad (8)$$

In order to see how SIA performs compared to SAA, compare (7) and (8) and observe that SIA results in a lower overhead than SAA if

$$\frac{1}{n}\sum_{x\in X}[1 - nq(x) - (1-q(x))^n]s(x) + s_K \sum_{x\in X} q(l(x)) \le 0.$$

This implies that SIA is better as long as

$$s_K \le \frac{\sum_{x\in X} \frac{nq(x)-1+(1-q(x))^n}{n} s(x)}{\sum_{x\in X} q(l(x))}.$$

The right-hand side of the above expression depends on n, the number of remote records, and is somewhat cumbersome to estimate. If possible, we would like to simplify it further. Now, it can be shown that, for all $q \in [0,1]$ and all $n \ge 1$,

$$\left(1-\frac{1}{n}\right)q^2 \le \frac{nq-1+(1-q)^n}{n} \le \left(1-\frac{1}{n}\right)q^2 + \frac{1}{4}.$$

In order to verify this, let $g(n) = \frac{nq-1+(1-q)^n-(n-1)q^2}{n}$; all we need to show is that $0 \le g(n) \le 0.25$. This is clearly true for $n = 1, 2$. Assuming that it is true for $n = m$, we get

$$(m-1)q^2 - mq + 1 \le (1-q)^m \le (m-1)q^2 - mq + 1 + \frac{m}{4}.$$

After substituting $(1-q)^m$ in the expression for $g(m+1)$ and performing the necessary algebraic manipulations, we get

$$\begin{aligned}
0 &\le \frac{q^2(m-1)(1-q)}{m+1} \le g(m+1)\\
&\le \frac{1}{4} - \frac{1}{4(m+1)}[mq(1-2q)^2 + 4q^2(1-q) + 1] \le \frac{1}{4}.
\end{aligned}$$

This completes the proof by induction. Therefore, SIA performs better as long as $s_K < \bar{s}_K$, where

$$\bar{s}_K = \frac{\left(1-\frac{1}{n}\right)\sum_{x\in X}[q(x)]^2 s(x)}{\sum_{x\in X} q(l(x))}.$$

Given a matching tree, it is now quite easy to estimate $\bar{s}_K$. In our experiments with real and synthetic data sets, we have found, except in the extreme case of $n = 1$, $\bar{s}_K$ to be much larger than s_K, thereby implying an overall superiority of SIA over SAA.

4.3 Concurrent Attribute Acquisition (CAA)

The main drawback of the sequential schemes (SAA and SIA) is that the information pertaining to the remote records must be transferred back and forth between the sites; the resulting overhead could be substantial, especially when the number of remote records is large. When we consider the latency-related delays as well, this back-and-forth nature of the communication may make them particularly inappropriate in many situations. This section proposes another approach that completely eliminates the overhead that occurs in a recursive partitioning scheme embedded in SAA or SIA. In this approach, we make use of the matching tree developed earlier to formulate a database query which is posed to the remote site to acquire only the relevant records. Although such a query would usually be quite long and complex, conceptually it is easily constructed by using the matching tree and can be generated automatically.

To understand how this can be done, first consider the vector $\mathcal{U} = (U_1, U_2, \ldots, U_K)$. Certain realizations of this vector leads to a matching probability greater than α; call these the *favorable* realizations. We are only interested in those remote records that lead to the favorable realizations of $\mathcal{U}$. We can use the tree effectively to identify these realizations. Any path in the tree that leads to a "STOP" node with a matching probability greater than α provides us with a favorable realization and is called an *acceptance path*. Such a path can be expressed as a conjunctive condition. For example, consider the acceptance path (Y_3, Y_7, Y_5, Y_9) in Fig. 1. This path can be expressed as a query condition:

$$(Y_3 = a(Y_3)) \ \wedge \ (Y_7 \neq a(Y_7)) \ \wedge (Y_5 = a(Y_5)) \\ \wedge (Y_9 = a(Y_9)).$$

Since many paths may lead to different favorable realizations, the overall query condition should be a disjunction of all the acceptance paths starting at the root. In other words, if there are m acceptance paths out of the root, and if e_j denotes the condition of path j, $j = 1, 2, \ldots, m$, then the overall query condition can be written as: $e_1 \vee e_2 \vee \cdots \vee e_m$. This query condition, however, is quite cumbersome (with many of the nodes repeated several times) and can be compressed further.

In order to explain how that can be done, denote by $E(x)$ the complete query condition rooted at node x. Because of the completeness property of the matching tree (as discussed in Section 3.2), every relevant record (a record with matching probability above α) must satisfy $E(x)$, and every irrelevant record (a record with matching probability below α) must satisfy $\neg E(x)$, the negation of $E(x)$. Let $l(x)$ and $r(x)$ be the left and right children of node x, respectively. Denoting the attribute at node x as $Y(x)$, $E(x)$ can be expressed as a recursion:

$$E(x) \equiv ((Y(x) = a(Y(x))) \wedge E(l(x))) \vee ((Y(x) \neq a(Y(x))) \\ \wedge E(r(x))).$$

Assume, without loss of generality, that a match on $Y(x)$ is a favorable realization. Now consider the revised query condition

$$E'(x) \equiv ((Y(x) = a(Y(x))) \wedge E(l(x))) \vee E(r(x)).$$

Clearly, $E(x) \Rightarrow E'(x)$, so a relevant record would not be excluded from consideration if $E(x)$ is replaced by $E'(x)$; the question is whether irrelevant records would be erroneously included as a result of this replacement. Suppose that $b \in R$ does not satisfy $E(x)$, i.e., b is irrelevant, but b satisfies $E'(x)$. Therefore, b must satisfy

$$E'(x) \wedge \neg E(x) \equiv (Y(x) = a(Y(x))) \wedge E(r(x)) \\ \wedge \neg E(l(x)) \Rightarrow (Y(x) = a(Y(x))) \wedge E(r(x)).$$

Since we assumed that a match on $Y(x)$ is a favorable realization, the matching probability of the above condition must be greater than the matching probability associated with the condition $(Y(x) \neq a(Y(x))) \wedge E(r(x))$. However, the latter condition corresponds to an acceptance path in the tree and has a matching probability greater than α, so b has a matching probability greater than α. This is a contradiction to the assumption that b is irrelevant. Therefore, by rewriting the expression of $E(x)$ as $E'(x)$ and using it recursively starting at the root, we can ensure each node is included in the query exactly once, thereby reducing the size of the query significantly. The size of the compressed query is, therefore, the size of tree and is equal to $\sum_{x \in X} s(x)$.

In this case, there is no identifier overhead, and the included record overhead is still $np|_{\emptyset} s_R$. Therefore, the normalized overhead in this case is given by

$$\text{Normalized Total Overhead} = \frac{\sum_{x \in X} s(x)}{n s_R} + p|_{\emptyset}. \qquad (9)$$

5 Testing and Validation

In order to study the actual performance of the above approaches, we implement and test them on real-world and synthetic data sets. Before we describe the implementation and discuss the results, two aspects of the numerical study should be discussed.

- The expected communication overhead for the sequential approach (normalized by the size of the remote database) can be calculated exactly based on the matching tree, as shown in (7), (8), and (9). Hence, we need not resort to simulation (using actual data sets) to estimate the expected communication overhead.
- The only role of the data set in this study is to estimate the conditional probabilities required for constructing the matching tree. As can be seen from (4), the probabilities necessary to construct the tree are: $p|_{\emptyset}$, $\Pr[U_k|M = 1]$, and $\Pr[U_k|M = 0]$, $k = 1, 2, \ldots, K$. Once these probabilities are estimated from the data set and stored, the matching tree can be easily constructed for different values of α. The communication overhead for each value of α can then be calculated.

5.1 Numerical Results

The first data set used in this study is a real-world database consisting of two versions of the MIS faculty directory maintained by the University of Minnesota—a 1992 version

(R_{92}) and a 1996 version (R_{96}). We designate the 1992 version as the "local" database and the 1996 version as the "remote" database. For training the model, we only use subsets of these databases with faculty records from the following randomly selected states: Alabama (AL), District of Columbia (DC), Maine (ME), Pennsylvania (PA), Illinois (IL), Oklahoma (OK), and New Jersey (NJ). There are 280 and 304 records in the 1992 (local) and 1996 (remote) subsets, respectively. There are two reasons as to why we consider records from a randomly selected subset of states, instead of randomly picking individual faculty records. First, this choice better mimics our problem context where the two databases have a significant overlap. If the individual faculty records were chosen randomly, the overlap would be much smaller (in fact, none in some cases). There is a secondary benefit as well—this choice yields much more accurate estimates of the probability parameters $\Pr[U_k|M = 1]$, $k = 1, 2, \ldots, K$. This is because, when records from the two databases are compared, only a few comparisons would result in a match ($M = 1$) if the overlap is small. However, if the overlap is large, there will be enough comparisons with $M = 1$, providing ample data points to estimate $\Pr[U_k|M = 1]$ accurately.

A related issue is that of the size of the training data set, which is always an important consideration in any empirical validation. In order to address this issue, we verify that the size of the data set used in our experiments is well above what is required to obtain accurate and stable estimates of the required probability parameters. This verification is done using the well-known χ^2 test—we find that the probability distribution remains unchanged even with data sets that are much smaller than the one used.

There are a total of 17 common attributes between R_{92} and R_{96}; the total size of the common attributes is 438 bytes per record. In order to establish the actual linkage, the records in the two versions are manually matched. We then obtain the necessary probabilities and construct the matching tree for different values of α. The computational overhead for this training phase is quite small. It takes well less than a minute to learn the probability parameters (Step 1 in Fig. 2) on a 3.2 GHz Pentium PC with 4 Gbytes of RAM, running Windows XP Professional (64-bit version). Once the probability parameters are learned, it takes only a few seconds to build a matching tree (Step 2 in Fig. 2).

Based on the tree, we calculate the communication overhead in each technique as a percentage of the size of the remote database. The results from the three different approaches are plotted in Fig. 4, for $s_K = 2$ bytes. This figure shows, for different values of α, the percentage overhead as a function of n, the number of remote records.

As can be clearly seen in this figure, the three tree-based approaches reduce the communication overhead significantly. Among these three, the overhead is the largest for SAA (hovering around 50-55 percent) and is independent of n. For our data set, SIA performed significantly better than SAA, except in the trivial extreme case of $n = 1$. The percentage overhead for SIA, however, quickly saturates to about 3-4 percent for values of n above 100. Since, in most real-world situations, the number of remote records is much larger than 100, SIA provides a very efficient strategy for record linkage. For still larger values of n (say, larger than about 10,000), an even more efficient strategy is provided by CAA. In this approach, the leading term of the normalized overhead asymptotically goes to zero with the number of remote records. The drawback with CAA is that the overhead could be high (even higher than the overhead needed to download the remote database) for smaller values of n (say, below 1,000).

The plots in Fig. 4 could be used to choose the right linkage strategy for a given situation. Since the tree can be precomputed, and the parameters of tree estimated, such plots could be generated for different values of α. At the time of actual usage, a simple database query could first be used to ascertain the number of remote records. An appropriate plot can then be consulted to determine which tree-based approach to use for obtaining the set of relevant records, with the least communication overhead.

Finally, it is interesting to note here that these tree-based approaches not only reduce the total communication overhead, but also requires considerably lower storage space for the local site and, therefore, is quite attractive for thin clients deployed for field work, for example, in crime investigation. Of the three approaches, in CAA, only the relevant records are sent to the local site, requiring only a small amount of storage. In SAA and SIA, however, one would need to locally store the identifier attribute values of all the remote records. While this is still manageable for smaller databases, this storage requirement could be prohibitively large for very large remote databases. However, as Fig. 4 clearly suggests, if the remote database is very large, we would prefer to use CAA which, as discussed above, requires only a small amount of storage.

5.2 Additional Validation

In addition to the numerical testing performed with the faculty directory database, we also test the tree-based approaches on a synthetic data set, to obtain additional insights and to strengthen the empirical validation. More specifically, we perform numerical tests using the *Freely Extensible Biomedical Record Linkage* (FEBRL) database—it is a large data repository containing synthetic census data and has been used in prior studies in record linkage [9], [18]. We use two synthetic data sets from FEBRL—the first data set contains 6,000 original records, and the second contains 4,000 synthetically generated duplicate records. For a record in the first data set, there maybe up to nine duplicates in the second data set. These duplicates are generated from the original data set by modifying attribute values—there are at most three modifications per attribute and at most 10 modifications per record. The first and the second data sets serve as the "local" and the "remote" data sets, respectively, in our experiments.

There are a total of 11 common attributes between the local and the remote databases; the total size of the common attributes is 196 bytes per record. Since most attribute values in the remote data set contain synthetically injected errors (to simulate typographical and other errors), an exact match of attribute values is quite rare in this case. As a result, if one insists on a perfect match, the matching patterns of attributes (U_k) do not provide sufficient information about matches in entities (M), and the subsequent record linkage is quite

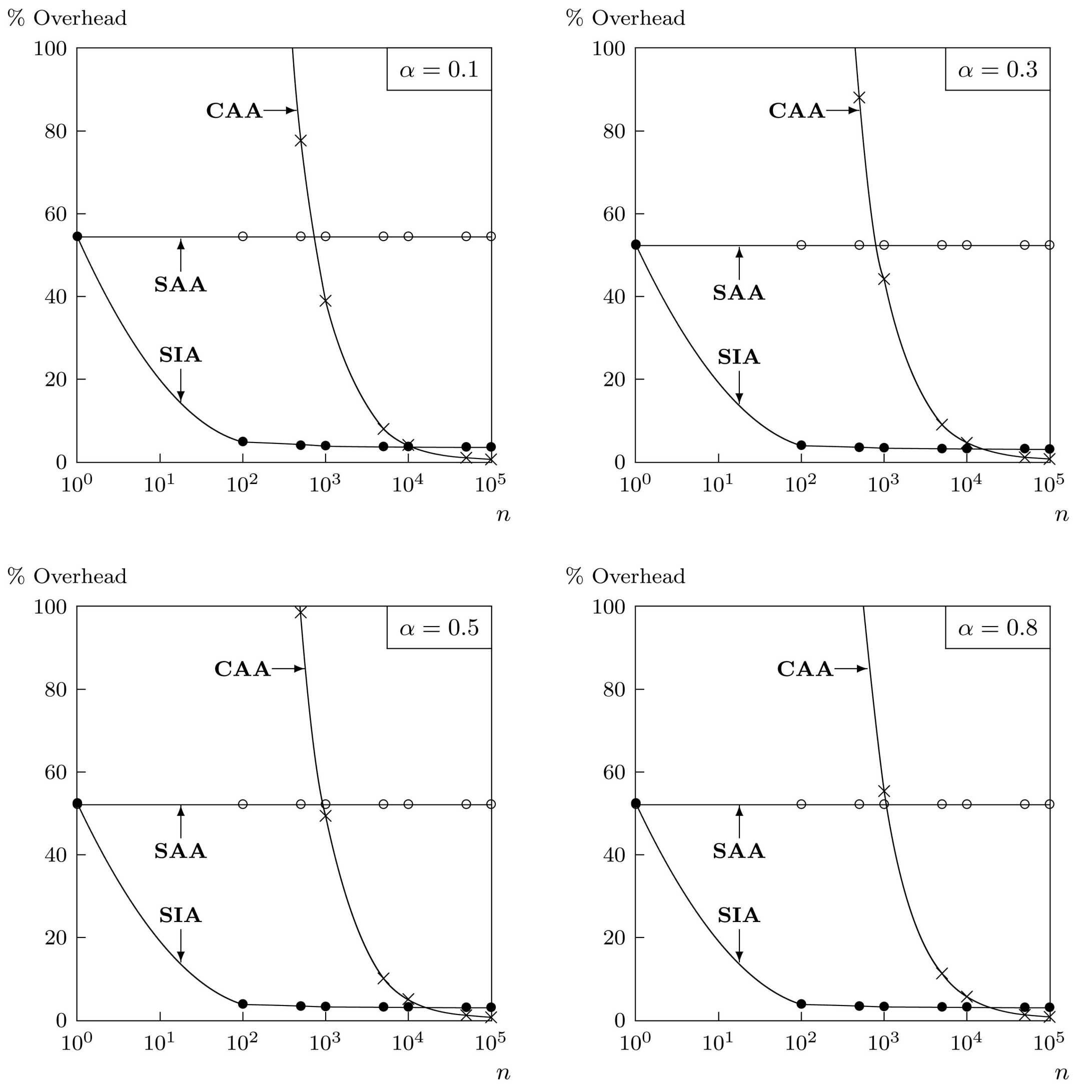


Fig. 4. Normalized overhead as a function of n in the three approaches.

error-prone. In order to circumvent this problem, we implement fuzzy matching for all the string-valued attributes. We first define a similarity measure between any two character strings, τ_1 and τ_2, based on a character-by-character comparison of these two strings:

$$\sigma(\tau_1, \tau_2) = \frac{1}{\mu}\sum_{i=1}^{\mu} I_{\tau_1[i]=\tau_2[i]},$$

where $I_{\tau_1[i]=\tau_2[i]}$ is 1 only if the ith characters of both the strings are the same, and it is zero otherwise; μ is the length of the shorter of the two strings. Now, given a string-valued attribute Y_k, and two records a and b, we redefine the matching pattern of Y_k as

$$U_k = \begin{cases} 1, & \text{if } \sigma(a(Y_k), b(Y_k)) \geq 0.8, \\ 0, & \text{otherwise.} \end{cases}$$

We run the three approaches on this data set; the results are shown in Fig. 5. Comparing Fig. 5 with Fig. 4, we find that, in terms of how the tree-based approaches contribute toward reducing the communication overhead, the results show similar trends. As before, SIA works better when the remote database is relatively small, and CAA, when the remote database is large. This provides additional validation of the efficacy of our approaches and their applicability in practice.

5.3 Matching Results

The focus so far has been the efficiency or the performance of the approaches in terms of reduction in communication overhead. We do not directly address the issue of the effectiveness or the performance of the approaches in terms of matching accuracy. Although we know from the completeness property of the tree that the matching performance of our approaches should be exactly the same as the full-information case, it would be nice to see how the tree-based approaches perform on real data.

To that end, we run two additional sets of experiments with our data sets. We first use the MIS faculty database. There, we select all the faculty records from the two versions (R_{92} and R_{96}) for the following states: Georgia (GA), Idaho (ID), Michigan (MI), New Hampshire (NH), Virginia (VA), Washington (WA), Wisconsin (WI), and California (CA). The

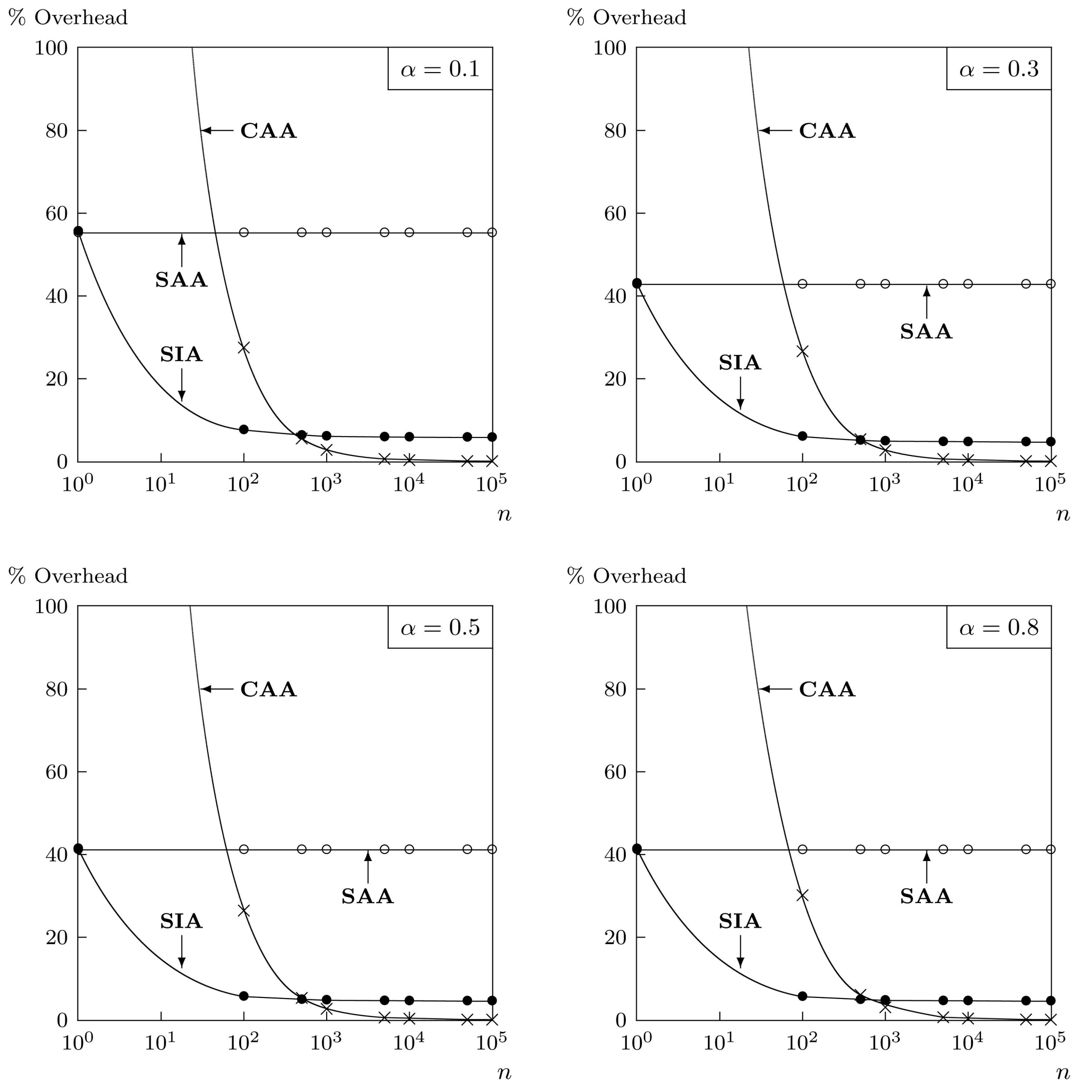


Fig. 5. Normalized overhead in the three approaches for the FEBRL data set.

1996 version with 497 records is designated as the remote database. We then run four trials with 50 queries each. Each query uses a randomly picked enquiry record (without substitution) from the local database. The results are provided in Table 1, in terms of the *recall* and *precision* measures. As can be clearly seen from this table, the tree-based approaches work very well in identifying the possible matches.

Next, we run basically the same experiment using the FEBRL data set. For this part of the experiment, the remote databases has 1,384 randomly picked records. The enquiry records are also picked randomly and a linkage query is run for each enquiry record. As before, we again run four trials, but this time with 100 queries each. The results are provided in Table 2. As can be seen from this table, the linkage performance is quite good, although not as good as the performance with the MIS faculty database (Table 1). This is not surprising. As mentioned earlier, the FEBRL database has a lot of attribute value modifications intentionally introduced to mimic typographical and other data entry and transformational errors that often creep into real-world databases. In fact, it is quite surprising that the linkage performance is this good, both in terms of *recall* and *precision*, despite all these errors. Overall, this shows the robustness of our approach and its flexibility to be adapted in different practical situations.

6 Conclusions

In this paper, we develop efficient techniques to facilitate record linkage decisions in a distributed, online setting. Record linkage is an important issue in heterogeneous database systems where the records representing the same real-world entity type are identified using different identifiers in different databases. In the absence of a common identifier, it is often difficult to find records in a remote database that are similar to a local enquiry record. Traditional record linkage uses a probability-based model to identify the closeness between records. The matching probability is computed based on common attribute values. This, of course, requires that common attribute values of all the remote records be transferred to the local site. The communication overhead is significantly large for such an operation. We propose techniques for record linkage that

TABLE 1
Matching Performance of Tree-Based Approaches (MIS Faculty Database)

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 1	32	0.1	100.0	100.0
		0.2	100.0	100.0
		0.3	100.0	100.0
		0.4	100.0	100.0
		0.5	100.0	100.0
		0.6	100.0	100.0
		0.7	100.0	100.0
		0.8	100.0	100.0
		0.9	100.0	100.0

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 2	25	0.1	100.0	96.2
		0.2	100.0	96.2
		0.3	100.0	100.0
		0.4	100.0	100.0
		0.5	100.0	100.0
		0.6	100.0	100.0
		0.7	100.0	100.0
		0.8	100.0	100.0
		0.9	100.0	100.0

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 3	39	0.1	100.0	97.5
		0.2	100.0	97.5
		0.3	100.0	97.5
		0.4	100.0	97.5
		0.5	100.0	97.5
		0.6	100.0	97.5
		0.7	100.0	97.5
		0.8	100.0	100.0
		0.9	100.0	100.0

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 4	32	0.1	100.0	94.1
		0.2	100.0	94.1
		0.3	100.0	94.1
		0.4	100.0	94.1
		0.5	100.0	94.1
		0.6	100.0	94.1
		0.7	100.0	97.0
		0.8	100.0	97.0
		0.9	100.0	97.0

TABLE 2
Matching Performance of Tree-Based Approaches (FEBRL Database)

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 1	91	0.1	100.0	89.2
		0.2	100.0	91.9
		0.3	97.8	96.7
		0.4	97.8	96.7
		0.5	97.8	97.8
		0.6	93.4	100.0
		0.7	93.4	100.0
		0.8	92.3	100.0
		0.9	92.3	100.0

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 2	57	0.1	98.3	86.2
		0.2	98.3	91.8
		0.3	98.3	94.9
		0.4	98.3	94.9
		0.5	98.3	96.6
		0.6	93.0	98.2
		0.7	93.0	98.2
		0.8	91.2	98.1
		0.9	91.2	98.1

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 3	98	0.1	100.0	89.1
		0.2	100.0	94.2
		0.3	99.0	98.0
		0.4	99.0	98.0
		0.5	98.0	99.0
		0.6	94.9	100.0
		0.7	94.9	100.0
		0.8	94.9	100.0
		0.9	94.9	100.0

Trial	*No. of true matches*	α	*Recall* (%)	*Precision* (%)
Trial 4	59	0.1	96.6	86.4
		0.2	96.6	86.4
		0.3	96.6	91.9
		0.4	96.6	91.9
		0.5	94.9	91.8
		0.6	94.9	100.0
		0.7	94.9	100.0
		0.8	94.9	100.0
		0.9	94.9	100.0

draw upon previous work in sequential decision making. More specifically, we develop a matching tree for attribute acquisition and propose three different schemes of using this tree for record linkage.

The three techniques proposed in this study were tested on real-world and synthetic data sets and were found to significantly reduce the communication overhead needed to perform record linkage. The worst of the three techniques still provided matching at a normalized overhead of about 40-50 percent (relative to the size of the remote database), whereas the other two provided the same results at a much lower overhead (less than 5 percent in most cases). Of these two, SIA did consistently well for all values of n, the number of remote records. CAA did not do well for low values of n, but at high values of n, it outperformed the other two techniques by a large margin. Based on these results, the appropriate technique could be adopted given a real-world application. Further, given the compelling nature of the results from our experiments, we expect our approach to work well in other real-world applications, too. It is necessary to point out that the proposed techniques reduce the communication overhead considerably, yet the linkage performance is assured to be at the same level as the traditional approach.

In this analysis, we compare three different approaches based on the expected size of communication overhead. One may also be interested in comparing these approaches in terms of transmission delay. Although we do not explicitly consider transmission delay, communication overhead is a reasonable surrogate, except in situations where network *latency* is a significant part of the delay. The introduction of latency into our analysis should further strengthen the case for CAA over the other approaches. Because of the sequential nature of the first two approaches, the total transmission there is broken into several smaller transmissions. Each of these smaller transmissions suffers from network latency. On the other hand, in CAA, the total transmission occurs in a single step—the query is sent once, and only one response is returned. Thus, the delay due to network latency is minimal in CAA.

There are several other directions for future research. For example, one could study how to extend the sequential model to cases where there are several enquiry records at the local site. In that case, the matching must be performed in a manner so that no local record is paired with more than one remote record and *vice versa*. The resulting sequential decision model would be different, and it may not be possible to build a decision tree a priori. Another avenue for future research is to perform an explicit cost-benefit trade-off between error cost and communication overhead. In this study, we were able to reduce the communication overhead significantly while keeping the error cost at the level of traditional techniques. It may, however, be possible to further reduce the communication overhead at the expense of incurring higher costs of linkage errors. One could also apply sequential decision-making techniques to the record linkage problem using nonprobabilistic similarity measures such as the distance-based measures used in clustering; this maybe useful in situations where the training data (to estimate the probabilities) are not readily available.

While discussing the efficiency of the tree-based techniques, we did not consider the extra computational load placed on the remote server. For SAA and SIA, the remote server would have to respond to a series of queries sent in a sequential manner. Especially for SIA, many of these queries would involve selecting records by comparing values of attributes for which a secondary index may not exist. The resulting load on the remote server may not always be insignificant. We did not consider this issue because the reduction of communication overhead is the major focus here. However, it would be interesting to study how the approaches can be implemented in a more efficient manner with respect to the load on the remote server.

Finally, an emerging area of research in record linkage concerns the preservation of privacy associated with the individual records shared across organizational boundaries. In this research, we consider situations where this is not a serious issue, or even if it is, it has been adequately addressed prior to applying our proposed approaches. However, it would be interesting to see if a linkage strategy such as the one proposed in this paper can make use of traditional privacy preservation ideas, such as data perturbation.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to the Associate Editor (Dr. Chengqi Zhang) and the three anonymous reviewers of *IEEE Transactions on Knowledge and Data Engineering* for their comments and suggestions; this paper has greatly benefited from their efforts.

REFERENCES

[1] R. Ash, *Information Theory.* John Wiley and Sons, 1965.

[2] J.A. Baldwin, "Linked Record Health Data Systems," *The Statistician,* vol. 21, no. 4, pp. 325-338, 1972.

[3] C. Batini, M. Lenzerini, and S.B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys,* vol. 18, no. 4, pp. 323-364, 1986.

[4] I. Baussano, M. Bugiani, D. Gregori, C. Pasqualini, V. Demicheli, and F. Merletti, "Impact of Immigration and HIV Infection on Tuberculosis Incidence in an Area of Low Tuberculosis Prevalence," *Epidemiology and Infection,* vol. 134, no. 6, pp. 1353-1359, 2006.

[5] R. Baxter, P. Christen, and T. Churches, "A Comparison of Fast Blocking Methods for Record Linkage," *Proc. ACM Workshop Data Cleaning, Record Linkage and Object Consolidation,* pp. 25-27, Aug. 2003.

[6] T.R. Belin and D.B. Rubin, "A Method for Calibrating False-Match Rates in Record Linkage," *J. Am. Statistical Assoc.,* vol. 90, no. 430, pp. 694-707, 1995.

[7] P. Bernstein, "Applying Model Management to Classical Meta Data Problems," *Proc. Conf. Innovative Database Research (CIDR),* pp. 209-220, Jan. 2003.

[8] J. Bischoff and T. Alexander, *Data Warehouse: Practical Advice from the Experts.* Prentice-Hall, 1997.

[9] P. Christen and T. Churches, *FEBRL: Freely Extensible Biomedical Record Linkage Manual,* Release 0.2 ed., https://sourceforge.net/projects/febrl/, Apr. 2003.

[10] J.B. Copas and F.J. Hilton, "Record Linkage: Statistical Models for Matching Computer Records," *J. Royal Statistical Soc.,* vol. 153, no. 3, pp. 287-320, 1990.

[11] M. DesMeules, J. Gold, S. McDermott, Z. Cao, J. Payne, B. Lafrance, B. Vissandjée, E. Kliewer, and Y. Mao, "Disparities in Mortality Patterns among Canadian Immigrants and Refugees, 1980-1998: Results of a National Cohort Study," *J. Immigrant Health,* vol. 7, no. 4, pp. 221-232, 2005.

[12] D. Dey, "Record Matching in Data Warehouses: A Decision Model for Data Consolidation," *Operations Research,* vol. 51, no. 2, pp. 240-254, 2003.

[13] D. Dey, S. Sarkar, and P. De, "A Probabilistic Decision Model for Entity Matching in Heterogeneous Databases," *Management Science,* vol. 44, no. 10, pp. 1379-1395, 1998.

[14] D. Dey, S. Sarkar, and P. De, "A Distance-Based Approach to Entity Reconciliation in Heterogeneous Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 14, no. 3, pp. 567-582, May/June 2002.

[15] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios, "Duplicate Record Detection: A Survey," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 1, pp. 1-16, Jan. 2007.

[16] I.P. Fellegi and A.B. Sunter, "A Theory of Record Linkage," *J. Am. Statistical Assoc.,* vol. 64, pp. 1183-1210, 1969.

[17] M.J. Goldacre, J.D. Abisgold, D.G.R. Yeates, and V. Seagroatt, "Risk of Multiple Sclerosis after Head Injury: Record Linkage Study," *J. Neurology, Neurosurgery, and Psychiatry,* vol. 77, no. 3, pp. 351-353, 2006.

[18] L. Gu and R. Baxter, "Adaptive Filtering for Efficient Record Linkage," *Proc. Fourth SIAM Int'l Conf. Data Mining (SDM '04),* pp. 22-24, Apr. 2004.

[19] L. Gu, R. Baxter, D. Vickers, and C. Rainsford, "Record Linkage: Current Practice and Future Directions," Technical Report 03/83, CSIRO Math. and Information Sciences, 2003.

[20] M.E. Hill, S.H. Preston, and I. Rosenwaike, "Age Reporting among White Americans Aged 85+: Results of a Record Linkage Study," *Demography,* vol. 37, no. 2, pp. 175-186, 2000.

[21] M.A. Jaro, "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," *J. Am. Statistical Assoc.,* vol. 84, no. 406, pp. 414-420, 1989.

[22] W. Kim, I. Choi, S. Gala, and M. Scheevel, "On Resolving Semantic Heterogeneity in Multidatabase Systems," *Distributed and Parallel Databases,* vol. 1, no. 3, pp. 251-279, 1993.

[23] M.D. Larsen and D.B. Rubin, "Iterative Automated Record Linkage Using Mixture Models," *J. Am. Statistical Assoc.,* vol. 96, no. 453, pp. 32-41, 2001.

[24] J.A. Larson, S.B. Navathe, and R. Elmasri, "A Theory of Attribute Equivalence in Databases with Application to Schema Integration," *IEEE Trans. Software Eng.,* vol. 15, no. 4, pp. 449-463, Apr. 1989.

[25] R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan, "Schema Equivalence in Heterogeneous Systems: Bridging Theory and Practice," *Information Systems,* vol. 19, no. 1, pp. 3-31, 1994.

[26] S.N. Minton, C. Nanjo, C.A. Knoblock, M. Michalowski, and M. Michelson, "A Heterogeneous Field Matching Method for Record Linkage," *Proc. Fifth IEEE Int'l Conf. Data Mining (ICDM '05),* pp. 314-321, Nov. 2005.

[27] J. Moore and A. Whinston, "A Model of Decision Making with Sequential Information Acquisition—Part I," *Decision Support Systems,* vol. 2, no. 4, pp. 285-307, 1986.

[28] J. Moore and A. Whinston, "A Model of Decision Making with Sequential Information Acquisition—Part II," *Decision Support Systems,* vol. 3, no. 1, pp. 47-72, 1987.

[29] H.B. Newcombe, M.E. Fair, and P. Lalonde, "The Use of Names for Linking Personal Records," *J. Am. Statistical Assoc.,* vol. 87, no. 420, pp. 1193-1204, 1992.

[30] H.B. Newcombe and J.M. Kennedy, "Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information," *Comm. ACM,* vol. 5, no. 11, pp. 563-566, 1962.

[31] C. Parent and S. Spaccapietra, "Issue and Approaches of Database Integration," *Comm. ACM,* vol. 41, no. 5es, pp. 166-178, 1998.

[32] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986.

[33] E. Rahm and P.A. Bernstein, "A Survey of Approaches to Automatic Schema Matching," *The VLDB J.,* vol. 10, no. 4, pp. 334-350, 2001.

[34] I. Schmitt and G. Saake, "A Comprehensive Database Schema Integration Method Based on the Theory of Formal Concepts," *Acta Informatica,* vol. 41, nos. 7/8, pp. 475-524, 2005.

[35] B. Tepping, "A Model for Optimum Linkage of Records," *J. Am. Statistical Assoc.,* vol. 63, pp. 1321-1332, 1968.

[36] A.M. Ward, N. de Klerk, D. Pritchard, M. Firth, and C.D. Holman, "Correlations of Siblings' and Mothers' Utilization of Primary and Hospital Health Care: A Record Linkage Study in Western Australia," *Social Science and Medicine,* vol. 62, no. 6, pp. 1341-1348, 2005.

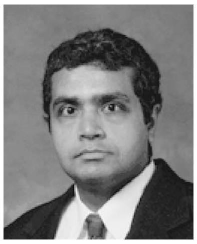
[37] W.E. Winkler, "Advanced Methods of Record Linkage," *Proc. Section Survey Research Methods,* pp. 467-472, 1994.

[38] W.E. Winkler, "Matching and Record Linkage," *Business Survey Methods,* B.G. Cox, D.A. Binder, B.N. Chinnappa, and A. Christianson, eds., Wiley & Sons, 1995.

[39] W.E. Winkler, "Methods for Evaluating and Creating Data Quality," *Information Systems,* vol. 29, pp. 531-550, 2004.

[40] D.S. Zingmond, Z. Ye, S. Ettner, and H. Liu, "Linking Hospital Discharge and Death Records-Accuracy and Sources of Bias," *J. Clinical Epidemiology,* vol. 57, no. 1, pp. 21-29, 2004.

Debabrata Dey received the PhD degree in computers and information systems from the University of Rochester. Currently, he is a professor of information systems at the University of Washington, Seattle. His current research interests are heterogeneous and distributed systems, network pricing and performance, information security, data warehousing, systems development, and contracting. His research has been published in *Management Science*, *Operations Research*, *Information Systems Research*, *ACM Transactions on Database Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *INFORMS Journal on Computing*, and several other leading journals and conference proceedings in Information Systems and Computer Science. He currently serves as a senior editor for *Information Systems Research* and as an associate editor for *Information Technology & Management*. In the past, he has served as associate editors for *Management Science*, *Information Systems Research*, *MIS Quarterly*, and *INFORMS Journal on Computing*. He is a member of the IEEE, the ACM, the INFORMS, and the AIS.



Vijay S. Mookerjee received the PhD degree in management, with a major in MIS, from Purdue University. Currently, he is a professor of information systems at the University of Texas at Dallas. His current research interests include social networks, optimal software development methodologies, storage and cache management, content delivery systems, and the economic design of expert systems and machine learning systems. He has published in and has articles forthcoming in several archival Information Systems, Computer Science, and Operations Research journals. He serves (or has served) on the editorial board of *Management Science*, *Information Systems Research*, *INFORMS Journal on Computing*, *Operations Research*, *Decision Support Systems*, *Information Technology & Management*, and *Journal of Database Management*.

Dengpan Liu received the PhD degree in management science with a concentration in information systems from the University of Texas at Dallas in 2006. He is currently an assistant professor of management information systems at the University of Alabama, Huntsville. His research interests include information security, software development, personalization at e-commerce sites, and data reconciliation.