Open access • Proceedings Article • DOI:10.1109/ICDE.2002.994694

TAILOR: a record linkage toolbox — Source link

Mohamed G. Elfeky, Vassilios S. Verykios, Ahmed K. Elmagarmid

Institutions: Purdue University

Published on: 07 Aug 2002 - International Conference on Data Engineering

Topics: System of record, Record linkage, Toolbox, Knowledge extraction and Data warehouse

TAILOR: A Record Linkage Toolbox*

Mohamed G. Elfeky [1] Vassilios S. Verykios [2] Ahmed K. Elmagarmid [1]
mgelfeky@cs.purdue.edu verykios@cis.drexel.edu ake@cs.purdue.edu
[1] Department of Computer Sciences, Purdue University, West Lafayette, IN.
[2] College of Information Science and Technology, Drexel University, Philadelphia, PA.

Abstract

Data cleaning is a vital process that ensures the quality of data stored in real-world databases. Data cleaning problems are frequently encountered in many research areas, such as knowledge discovery in databases, data warehousing, system integration and e-services. The process of identifying the record pairs that represent the same entity (duplicate records), commonly known as record linkage, is one of the essential elements of data cleaning. In this paper, we address the record linkage problem by adopting a machine learning approach. Three models are proposed and are analyzed empirically. Since no existing model, including those proposed in this paper, has been proved to be superior, we have developed an interactive ***R****ec***o***rd* ***Li****nk***a***ge* ***T****oolbox named TAILOR. Users of TAILOR can build their own record linkage models by tuning system parameters and by plugging in in-house developed and public domain tools. The proposed toolbox serves as a framework for the record linkage process, and is designed in an extensible way to interface with existing and future record linkage models. We have conducted an extensive experimental study to evaluate our proposed models using not only synthetic but also real data. Results show that the proposed machine learning record linkage models outperform the existing ones both in accuracy and in performance.*

1. Introduction

Record linkage is the process of comparing the records from two or more data sources in an effort to determine which pairs of records represent the same real-world entity. Record linkage may also be defined as the process of discovering the duplicate records in one file. What makes record linkage a problem in its own right, (i.e., different from the duplicate elimination problem [2]), is the fact that real-world data is "dirty". In other words, if data were accurate, record linkage would be similar to duplicate elimination, since the duplicate records would have the same values in all fields. Yet, in real-world data, duplicate records may have different values in one or more fields. For example, more than one record may correspond to the same person in a customer database because of a misspelled character in the name field. Record linkage is related to the similarity search problem, which is concerned with the retrieval of those objects that are similar to a query object. In particular, record linkage may use similarity search techniques in order to search for candidate similar records. From these candidate similar records, record linkage should determine only those that are actually duplicates.

Record linkage can be considered as part of the *data cleansing* process, which is a crucial first step in the knowledge discovery process [9]. In 1969, Fellegi and Sunter [10] were the first to introduce the formal mathematical foundations for record linkage, following a number of experimental papers that were published since 1959 [25]. The model proposed by Fellegi and Sunter, which is briefly discussed in Section 2.2, is characterized as a probabilistic model since it is entirely based on probability theory. Winkler [34] surveys the research that extends and enhances the model proposed by Fellegi and Sunter.

The record linkage problem can be viewed as a pattern classification problem. In pattern classification problems, the goal is to correctly assign patterns to one of a finite number of classes. By the same token, the goal of the record linkage problem is to determine the matching status of a pair of records brought together for comparison. Machine learning methods, such as decision tree induction, neural networks, instance-based learning, clustering, etc., are widely used for pattern classification. Specifically, given a set of patterns, a machine learning algorithm builds a model that can be used to predict the class of each unclassified pattern. Machine learning methods are categorized into two main groups: supervised learning and unsupervised learning. A method is supervised if a training set is available; otherwise the method is unsupervised [22]. Cochinwala et al. [5], and Verykios et al. [32] were the first to exploit the use of decision tree induction for the solution of the record linkage problem.

1.1 Contributions

The first contribution of this paper is the development of a **R**ec**o**rd **Li**nk**a**ge **T**oolbox (TAILOR) that can be tailored to fit any record linkage model. TAILOR implements state-of-the-art tools and models for linking records. Since none of the proposed record linkage models has been presented as the best one, the development of

* This research is partially supported by NSF under grant 9972883-EIA, 9974255-IIS, and 9983249-EIA, and by grants from IBM, NCR, Telcordia, and Wal-Mart.

such a toolbox is significant.

A new machine learning approach for the record linkage problem is the second contribution of this paper. The introduction of such an approach raises the limitations of previous record linkage models, which can handle only binary or categorical comparisons. Three machine learning record linkage models are proposed: an induction model, a clustering model and a hybrid model.

The third contribution is the extensive experimental study that analyzes and compares the record linkage models and tools using synthetic data, generated by a public domain tool (DBGen), as well as real data from a Wal-Mart database. Towards this end, we have proposed novel accuracy and performance metrics. The empirical results show that our proposed machine learning record linkage models outperform the probabilistic record linkage model with respect to most performance and accuracy metrics.

1.2 Paper Organization

The rest of this paper is organized as follows. In Section 2 the record linkage problem is introduced along with the notation that is used throughout the paper. Moreover, a brief discussion of the probabilistic record linkage model proposed by Fellegi and Sunter [10] is given. In Section 3, we present the newly developed machine learning models for the record linkage problem. Section 4 discusses the system architecture of the record linkage toolbox, along with a brief discussion of the tools, which we developed. In Section 5, a large number of experiments are conducted. In Section 6, we summarize other related work, and finally we conclude our study in Section 7.

2. Record Linkage Problem

2.1 Definition and Notation

For two data sources *A* and *B*, the set of ordered record pairs $AXB = \{(a,b) : a \in A, b \in B\}$ is the union of two disjoint sets, *M* where $a = b$ and *U* where $a \neq b$. We call the former set *matched* and the latter set *unmatched*. The problem, then, is to determine in which set each record pair belongs to. Having in mind that it is always better to classify a record pair as a possible match than to falsely decide on its matching status with insufficient information, a third set *P*, called *possible matched*, is introduced. In the case that a record pair is assigned to *P*, a domain expert should manually examine this pair. We assume that a domain expert can always identify the correct matching status (*M* or *U*) of a record pair.

Let us assume that a record obtained from either source *A* or source *B* contains *n* components (fields), $f_1, f_2, \ldots, f_n$. For each record pair $r_{i,j} = (r_i, r_j)$, the component-wise comparison results in a vector of *n* values, $c_{i,j} = [\, c_1^{i,j}, c_2^{i,j}, \ldots, c_n^{i,j} \,]$ such that $c_k^{i,j} = C_k(r_i.f_k, r_j.f_k)$ and C_k is the comparison function that compares the values of the record component *k*. The resulting vector is called a *comparison vector*. The set of all the comparison vectors is called the *comparison space*. A comparison function C_k is a mapping from the Cartesian product of the domain(s) for the field f_k to a comparison domain R_k; formally, $C_k : D_k \times D_k \to R_k$. Examples of simple comparison functions are

$$C_I(value_1, value_2) = \begin{cases} 0 & \text{if } value_1 = value_2 \\ 1 & \text{otherwise} \end{cases}, \text{ and}$$

$$C_{II}(value_1, value_2) = \begin{cases} 0 & \text{if } value_1 = value_2 \\ 1 & \text{if } value_1 \text{ or } value_2 \text{ is missing} \\ 2 & \text{otherwise} \end{cases},$$

where $R_I = \{0,1\}$, and $R_{II} = \{0,1,2\}$. The value computed by C_I is called a *binary comparison value*, while this computed by C_{II} is called a *categorical comparison value*. The *continuous comparison value* is another type that is computed by comparison functions that are based on a distant metric between the two compared values. More complex comparison functions will be presented in Section 4.1.2.

2.2 (Error-Based) Probabilistic Record Linkage Model

For each record pair $r_{i,j}$, let us define m_k and u_k as: $m_k = \text{Prob}\{c_k^{i,j} = 0 \mid r_{i,j} \in M\}, u_k = \text{Prob}\{c_k^{i,j} = 0 \mid r_{i,j} \in U\}$. By denoting $\text{Prob}\{r_{i,j} \in M\}$ as $\text{Prob}\{M \mid r_{i,j}\}$, and similarly $\text{Prob}\{r_{i,j} \in U\}$ as $\text{Prob}\{U \mid r_{i,j}\}$, and by assuming that the independence assumption holds, we can derive the following: $\text{Prob}\{r_{i,j} \mid M\} = \prod_{k=1}^{n} m_k^{c_k^{i,j}} (1 - m_k)^{1-c_k^{i,j}}$, and $\text{Prob}\{r_{i,j} \mid U\} = \prod_{k=1}^{n} u_k^{c_k^{i,j}} (1 - u_k)^{1-c_k^{i,j}}$. The probabilistic record linkage model defined by Fellegi and Sunter [10] assigns a weight $w_k^{i,j}$ for each component of each record pair, that is

$$w_k^{i,j} = \begin{cases} log(m_k / u_k) & \text{if } c_k^{i,j} = 0 \\ log((1 - m_k)/(1 - u_k)) & \text{if } c_k^{i,j} = 1 \end{cases}.$$

A decision is made for each record pair by calculating a composite weight $L(r_{i,j}) = \sum_{k=1}^{n} w_k^{i,j}$, and by comparing this value against two threshold values $t_1 < t_2$, that is $r_{i,j} \in M$ if $L(r_{i,j}) \geq t_2$, $r_{i,j} \in U$ if $L(r_{i,j}) \leq t_1$, and

$r_{i,j} \in P$ if $t_1 < L(r_{i,j}) < t_2$. The issue is to determine estimates of the conditional probabilities m_k and u_k for $k = 1,2,\ldots,n$, as well as estimates of the thresholds t_1 and t_2. Although the probabilistic record linkage model is presented in such a way that it considers only binary comparison values, it can be adjusted to support categorical comparison values as well [34].

The thresholds t_1 and t_2 can be estimated by minimizing the probability of the error of making an incorrect decision for a record pair [18]; this is the reason why the model is called error-based. In practice, the record pairs are sorted in ascending order of their composite weight, and indexed according to this order $r_1, r_2, \ldots r_N$ where N is the size of the comparison space. The maximum weight for an unmatched record pair is the weight of the record pair $r_{N'}$ where $\sum_{l=1}^{N'} \text{Prob}\{r_l \mid M\} \le p_1$ and p_1 is the acceptable error probability of misclassifying a matched record pair as unmatched. The minimum weight for a matched record pair is the weight of the record pair $r_{N''}$ where $\sum_{l=N''}^{N} \text{Prob}\{r_l \mid U\} \le p_2$ and p_2 is the acceptable error probability of misclassifying an unmatched record pair as matched. Fellegi and Sunter in [10] proved that this decision procedure is optimal.

Fellegi and Sunter proposed two methods for estimating the conditional probabilities m_k and u_k for $k = 1,2,\ldots,n$. A different approach, explored in [33], uses the EM (Expectation Maximization) method [6]. The latter approach is proved to be very effective since it is highly stable and the least sensitive to initial values [18].

2.3 EM-Based Probabilistic Record Linkage Model

The EM algorithm considers the estimation of a family of parameters ϕ for a data set *x* given an incomplete version of this data set *y*. By postulating a family of sampling densities $f(x \mid \phi)$ and deriving its corresponding family of sampling densities $h(y \mid \phi)$, the EM algorithm is directed to find a value of ϕ which maximizes $h(y \mid \phi)$. A detailed description of the EM algorithm can be found in [6].

In the probabilistic record linkage model, the parameters to estimate are $\phi = (m_1, m_2, \ldots, m_n, u_1, u_2, \ldots, u_n, p)$ where p is the proportion of the matched record pairs $|M|/N$ and N is the total number of record pairs. The whole set of comparison vectors is considered to be the incomplete data set *y*. The missing part from each comparison vector $c_l = [c_1^l, c_2^l, \ldots, c_n^l]$, denoted as g_l, for $l = 1,2,\ldots,N$, corresponds to whether this comparison vector represents a *matched* record pair or an *unmatched* pair, that is $g_l = [1,0]$ if c_l represents a *matched* record pair, and $g_l = [0,1]$ if c_l represents an *unmatched* record pair. The complete data log-likelihood is

$$\begin{aligned} ln\, f(y \mid \phi) = &\sum_{l=1}^{N} g_l \cdot (ln\,\text{Prob}\{c_l \mid M\}, ln\,\text{Prob}\{c_l \mid U\})^T \\ &+ \sum_{l=1}^{1N} g_l \cdot (ln\, p, ln(1-p))^T \end{aligned}.$$

Given a set of initial values for the unknown parameters, the EM algorithm applies several expectation and maximization iterations until the desired precision of the estimated values is obtained. In the expectation step, g_l is replaced by $(g_m(c_l), g_u(c_l))$ where

$$g_m(c_l) = \frac{p \prod_{k=1}^{n} m_k^{c_k^l} (1-m_k)^{1-c_k^l}}{p \prod_{k=1}^{n} m_k^{c_k^l} (1-m_k)^{1-c_k^l} + (1-p) \prod_{k=1}^{n} u_k^{c_k^l} (1-u_k)^{1-c_k^l}}$$

and $g_u(c_l)$ can be derived similarly for each $l = 1,2,\ldots,N$. In the maximization step, the data log-likelihood can be separated into three maximization problems. By setting the partial derivatives equal to 0, we obtain the values of the unknown parameters:

$$m_k = \frac{\sum_{l=1}^{N} c_k^l \cdot g_m(c_l)}{\sum_{l=1}^{N} g_m(c_l)},\ u_k = \frac{\sum_{l=1}^{N} c_k^l \cdot g_u(c_l)}{\sum_{l=1}^{N} g_u(c_l)},\ p = \frac{\sum_{l=1}^{N} g_m(c_l)}{N}.$$

2.4 Cost-Based Probabilistic Record Linkage Model

The thresholds t_1 and t_2 are estimated by minimizing the probability of the error of making an incorrect decision for the matching status of a record pair. In practice, the minimization of the probability of the error is not the best criterion to use in designing a decision rule as different wrong decisions may have different consequences. For example, the incorrect decision to classify an unmatched record pair in the matched set may lead to an undesired action of removing one of the records, whereas the incorrect decision to classify a matched record pair as unmatched may lead to data inconsistencies. Based on the above observations, a cost-based probabilistic record linkage model that is currently being developed by the authors [30] is important.

3. Machine Learning Approach

One of the disadvantages of the probabilistic record linkage model is its ability to handle only binary or categorical comparison vector attributes. Our goal is to overcome this disadvantage using new machine learning approach. The proposed machine learning record linkage models can handle all comparisons types, including the continuous ones. Another disadvantage of the probabilistic record linkage model is that it relies on the existence of a training set. Although the proposed induction record linkage model has the same disadvantage, both the clustering and the hybrid record linkage models do not.

3.1 Induction Record Linkage Model

In supervised machine learning, a training set of patterns in which the exact class of each pattern is known apriori, is used in order to build a classification model that can be used afterwards to predict the class of each unclassified pattern. A training instance has the form $< x, f(x) >$ where *x* is a pattern, and $f(x)$ is a discrete-valued function that represents the class of the pattern *x*, i.e., $f(x) \in \{L_1, L_2, \ldots, L_m\}$ where *m* is the number of the possible classes. The classification model can be defined as an approximation to *f* that is to be estimated using the training instances. A supervised learning technique can be called a *classifier*, as its goal is to build a classification model. Induction of decision trees [27] and instance-based learning [1], which are called inductive learning techniques, are two examples of classifiers. These techniques share the same approach to learning. This approach is based on exploiting the regularities among observations, so that predictions are made on the basis of similar, previously encountered situations. The techniques differ, however, in the way of how similarity is expressed: decision trees make important shared properties explicit, whereas instance-based techniques equate (dis)similarity with some measure of distance. By itself, the induction of decision trees technique does feature selection that decreases the cost of prediction.

The proposed induction record linkage model is illustrated in Figure 1. The training set consists of instances of the form $< c, f(c) >$ where *c* is a comparison vector and $f(c)$ is its corresponding matching status, i.e., $f(c) \in \{M, U\}$ where *M* denotes a matched record pair and *U* denotes an unmatched one. A classifier is employed to build a classification model that estimates the function *f* and is able to predict the matching status of each comparison vector of the whole set of record pairs. Observe that *P* is not included in the domain of $f(c)$ based on the assumption in Section 2.1, and the fact that the training instances are obtained by a domain expert.

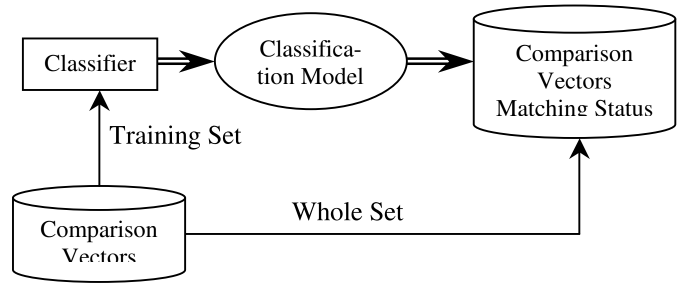


Figure 1. Induction Record Linkage Model

3.2 Clustering Record Linkage Model

The disadvantage of the previous model, as well as of the probabilistic record linkage model, is that it relies on the existence of a training set. Such a training set is not readily available for most real-world applications. In unsupervised learning methods, the notion of a training set does not exist. The whole set of patterns is given as input to the unsupervised learning algorithm to predict the class of each unclassified pattern, or in the record linkage case, the matching status of each record pair. Following the same notation used in the previous section, unsupervised learning tries to approximate the function *f* without having any training instances. Clustering is the only known way for unsupervised learning, and so the model proposed can be called *clustering record linkage model*. The fundamental clustering problem involves grouping together those patterns that are similar to each other [3]. In other words, if each pattern is represented as a point in the space, clustering algorithms try to cluster these points into separate groups in the space. A specific technique, called *k-means clustering*, tries to cluster the points into *k* clusters. This technique is used specifically when the number of classes of the data items is known

The clustering record linkage model considers each comparison vector as a point in *n*-dimensional space, where *n* is the number of components in each record. A clustering algorithm, such as *k-means clustering*, is used to cluster those points into three clusters, one for each possible matching status, *matched*, *unmatched*, and *possibly matched*. After applying the clustering algorithm to the set of comparison vectors, the issue is to determine which cluster represents which matching status.

Let $c_{i,j} = [c_1^{i,j}, c_2^{i,j}, \ldots, c_n^{i,j}]$ be the comparison vector resulting from component-wise comparison of the two records r_i, r_j. Assuming that all the comparison functions are defined in such a way that the value 0 means a perfect agreement between the two compared values, then $c_k^{i,j} = 0$ means that the two compared values $r_i.f_k$ and $r_j.f_k$ agree perfectly. Therefore, a perfectly matched record pair that agrees in all fields results in a comparison vector that has zeros in all of its components, i.e., its location coincides with the origin in n-dimensional space. Similarly, a completely unmatched record pair results in a comparison vector that has 1's in all its components. Hence, in order to determine which cluster represents which matching status, the central point of each cluster in the space is determined. The nearest cluster to the origin is considered to be the cluster that represents the matched record pairs, whereas the farthest cluster from the origin is considered to be the one that represents the unmatched record pairs. The remaining cluster is considered the one that represents the possibly matched record pairs.

3.3 Hybrid Record Linkage Model

The third model proposed in this paper is the hybrid record linkage model. Such a model combines the advantages of both the induction and the clustering record linkage models. Supervised learning gives more accurate results for pattern classification than unsupervised learning. However, supervised learning relies on the presence of a training set, which is not available in practice for many applications. Unsupervised learning can be used to overcome this limitation by applying the unsupervised learning on a small set of patterns in order to predict the class of each unclassified pattern, i.e., a training set is generated.

The proposed hybrid record linkage model proceeds in two steps. In the first step, clustering is applied to predict the matching status of a small set of record pairs. A training set is formed as $\{< c, f(c) >\}$ where c is a comparison vector and $f(c)$ is the predicted matching status of its corresponding record pair, i.e., $f(c) \in \{M, U, P\}$ where P denotes a possible matched record pair, and M and U are as before. In the second step, a classifier is employed to build a classification model just like the induction record linkage model.

4. Record Linkage Toolbox TAILOR

TAILOR is a record linkage toolbox that can be used to build a complete record linkage model by tuning a few parameters and plugging in some in-house developed and public domain tools. It encompasses all tools and models proposed thus far in the literature for solving the record linkage problem, and includes performance and accuracy metrics to compare these different models.

4.1 System Design

The record linkage process comprises two main steps. The first step is to generate the comparison vectors by component-wise comparison of each record pair. The second step is to apply the decision model to the comparison vectors to determine the matching status of each record pair. Figure 2 shows the layered design of TAILOR.

Graphical User Interface
Measurement Tools
Decision Models
Comparison Functions
Searching Methods
Database Management System

Figure 2. TAILOR Layered Design

In the bottom layer of the system is the database management system itself, through which data is accessed. The topmost layer is a graphical user interface so that the toolbox can be easily used. Between the database and the graphical user interface, TAILOR contains four layers: S*earching Methods*, *Comparison Functions*, *Decision Models* and *Measurement Tools*. Table 1 gives a complete list of the various models and tools implemented in each layer.

Searching Methods	- Blocking
	  - Sorting
	  - Hashing
	- Sorted Neighborhood

Comparison Functions	- Hamming Distance - Edit Distance - Jaro's Algorithm - N-grams - Soundex Code

Decision Models	- Probabilistic Model
	  - EM-Based
	  - Cost-Based
	  - Error-Based
	- Induction Model
	- Clustering Model
	- Hybrid Model

Measurement Tools	- Reduction Ratio - Pairs Completeness - Accuracy - Completeness

Supporting Tools	- MLC++
	  - ID3 decision trees
	  - IBL instance-based learning
	- DBGen

Table 1. TAILOR Tools List

Figure 3 shows the information flow diagram between these four layers. It shows how the record linkage process operates. First, a searching method is exploited to reduce the size of the comparison space. It is very expensive to consider all possible record pairs for comparison. For a data file of n records, the number of record pairs that can be generated is equal to $n(n-1)/2$, i.e., $O(n^2)$. In order to reduce the large space of record pairs, searching methods are needed to select a smaller set of record

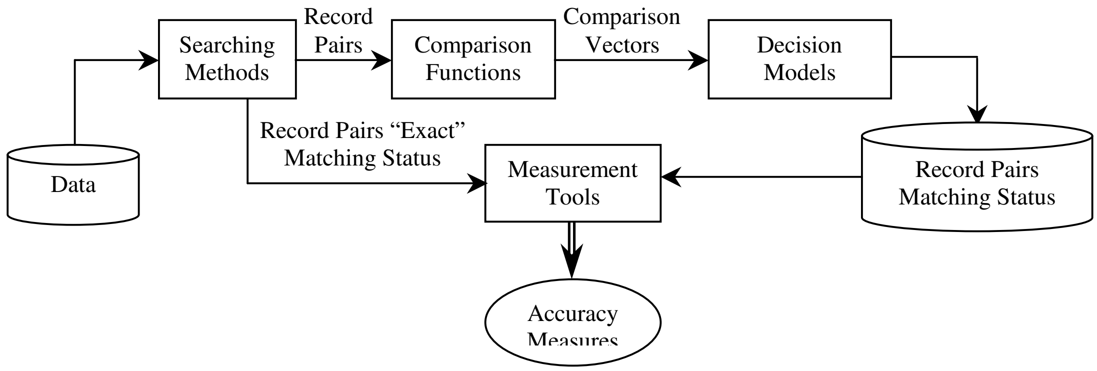


Figure 3. TAILOR Information Flow Diagram

pairs that are candidates to be matched. They should be intelligent enough to exclude any record pair whose two records completely disagree, i.e., to exclude any record pair that cannot be a potentially matched pair. The selected record pairs are provided to the comparison functions to perform component-wise comparison of each record pair, and hence generate the comparison vectors. Then, the decision model is applied to predict the matching status of each comparison vector. Last, an evaluation step, to estimate the performance of the decision model, is performed.

4.1.1 Searching Methods

4.1.1.1 Blocking

Blocking is defined as a partition of the file into mutually exclusive blocks [24]. Comparisons are restricted to records within each block. Blocking can be implemented by sorting the file according to a block key [18]. A block key is a combination of one or more record fields, or portions of them. The records that agree in the block key are assigned to the same block. A more efficient way to implement blocking is by using hashing. A record is hashed according to its block key in a hash block. Only records in the same hash block are considered for comparison.

The number of generated record pairs depends on the number of blocks, which subsequently depends on the block key. In order to have some insight into the size of this number, let b be the number of blocks, and assume that each block has n/b records. The number of record pairs will be $b \cdot O(n^2/b^2)$, that is $O(n^2/b)$. The total time complexity of blocking is $O(h(n)+n^2/b)$ where $h(n)=n\log n$ if blocking is implemented using sorting, or $h(n)=n$ if blocking is implemented using hashing.

4.1.1.2 Sorted Neighborhood

The Sorted Neighborhood method, discussed in [15], sorts the data file first, and then moves a window of a specific size w over the data file, comparing only the records that belong to this window. In this way, the maximum number of comparisons for each record is reduced to $2w-1$. Several scans, each of which uses a different sorting key, may be applied to increase the possibility of combining matched records.

An analysis for the time complexity of this method is found in [15]. The sorting phase requires $O(n\log n)$. The number of record pairs, generated by the sorted neighborhood method of window size w, is $(w-1)(n-w/2)$, which is $O(wn)$. Thus, the total time complexity is $O(n\log n + wn)$.

4.1.2 Comparison Functions

4.1.2.1 Hamming Distance

The Hamming distance is used primarily for numerical fixed size fields like Zip Code or SSN. It counts the number of mismatches between two numbers. For example, the Hamming distance between zip codes "47905" and "46901" is 2 since it has 2 mismatches.

4.1.2.2 Edit Distance

The Hamming distance function cannot be used for variable length fields since it does not take into account the possibility of a missing letter, e.g., "John" and "Jon", or an extra letter, e.g., "John" and "Johhn". The edit distance between two strings is the minimum cost to convert one of them to the other by a sequence of character insertions, deletions, and replacements. Each one of these modifications is assigned a cost value. For example, if we assume that the insertion cost and the deletion cost are each equal to 1, and the replacement cost is equal to ∞, then the edit distance between "John" and "Jon" is 1, and the edit distance between "John" and "Jonn" is 2. In order to achieve reasonable accuracy, the modifications costs should be tuned specifically for each string data set. Zhu and Ungar [35] use genetic algorithms to learn these costs. An efficient algorithm to compute the edit distance is the Smith-Waterman algorithm [29] that uses a dynamic programming technique.

4.1.2.3 Jaro's Algorithm

Jaro [17] introduced a string comparison function that accounts for insertions, deletions, and transpositions. Jaro's algorithm finds the number of common characters and the number of transposed characters in the two strings. A common character is a character that appears in both strings within a distance of half the length of the shorter string. A transposed character is a common character that appears in different positions. For example, comparing "John" to "Jhon" results in four common characters, two of which are transposed, while comparing "John" to "Jon" results in three common characters, none of which is transposed. The value of Jaro's comparison is defined as $\left(c/l_1 + c/l_2 + (2c-t)/2c\right)/3$, where c is the number of common characters, t is the number of transposed characters, and l_1, l_2 are the lengths of the two strings.

4.1.2.4 N-grams

N-grams is another approach for computing the distance between two strings. The N-grams comparison function forms the set of all the substrings of length n for each string. The distance between the two strings is defined as $\sqrt{\sum_{\forall x} |f_a(x) - f_b(x)|}$ where $f_a(x)$ and $f_b(x)$ are the number of occurrences of the substring x in the two strings a and b, respectively. Bigrams comparison ($n = 2$) is known to be very effective with minor typographical errors. It is widely used in the field of information retrieval [11]. Trigrams comparison ($n = 3$) is used by Hylton [16] in record linkage of bibliographical data. Most recently, N-grams was extended to what is referred to as Q-grams [13] for computing approximate string joins efficiently. N-grams is more efficient than edit distance or Jaro's algorithm in the case of strings that contain multiple words and are known to be commonly in error with respect to word order. For example, comparing "John Smith" with "Smith John" results in 0.342 using Jaro's algorithm, 0.5 using edit distance, 0.375 using trigrams, 0.222 using bigrams. Bigrams comparison gives the lowest value, which means that the two strings are much closer using bigrams than using other comparison functions.

4.1.2.5 Soundex Code

The purpose of the Soundex code is to cluster together names that have similar sounds [19]. For example, the Soundex code of "Hilbert" and "Heilbpr" is similar; as is the Soundex code of "John" and "Jon". The Soundex code of a name consists of one letter followed by three numbers. The letter is the first letter of the name. Disregarding the remaining vowels, as well as the letters W, Y and H, the numbers are assigned to the first three letters following the first letter according to Table 2. An exception is when two letters that have the same number occur consecutively. In the latter case, the second letter is ignored. The Soundex code is padded by 0's if less than three numbers are encountered. For example, the Soundex code for both "Hilbert" and "Heilbpr, is H416; the Soundex code for both "John" and "Jon" is J500.

Letters	Number	Letters	Number
B, F, P, V	1	C, G, J, K, Q, S, X, Z	2
D, T	3	L	4
M, N	5	R	6

Table 2. Soundex Code Guide

4.1.3 Measurement Tools

TAILOR provides several performance metrics, some of which were proposed in a previous study [31]. The following subsections briefly introduce these metrics using the following notation. Let n_M and n_U be the total number of matched and unmatched record pairs in the entire data, respectively. Let s be the size of the reduced comparison space generated by the searching method, and let s_M and s_U be the number of matched and unmatched record pairs in this reduced comparison space, respectively. Finally, let $c_{a,d}$ be the number of record pairs whose actual matching status is a, and whose predicted matching status is d, where a is either M or U, and d is either M, U or P, where M, U and P represent the matched, unmatched and possibly matched, respectively.

4.1.3.1 Reduction Ratio

The *reduction ratio* metric is defined as $RR = 1 - s/(n_M + n_U)$. It measures the relative reduction in the size of the comparison space accomplished by a searching method.

4.1.3.2 Pairs Completeness

A searching method can be evaluated based on the number of actual matched record pairs contained in its reduced comparison space. We define the *pairs completeness* metric as the ratio of the matched record pairs found in the reduced comparison space, to the total number of matched record pairs in the entire comparison space. Formally, the *pairs completeness* metric is defined as $PC = s_M / n_M$.

4.1.3.3 Accuracy

The *accuracy* metric tests how accurate a decision model is. The *accuracy* of a decision model is defined to be the percentage of the correctly classified record pairs. Formally, the *accuracy* metric is defined as $AC = (c_{M,M} + c_{U,U})/s$.

4.1.3.4 Completeness

The *completeness* metric tests how complete the decision model is when considering the matched record pairs. The *completeness* metric is defined as the ratio of the matched record pairs detected by the decision model to the total number of matched record pairs known in the data. The *completeness* metric cannot be expressed as a function of the previously introduced terms since transitivity is taken into consideration while computing this metric. Transitivity means that if record *x* matches record *y* and record *y* matches record *z*, then the record pair (*x*, *z*) should be declared as matched even if it has another predicted matching status.

4.1.4 Supporting Tools

TAILOR incorporates other ready-made tools in order to provide additional functionality. The first one is MLC++ [20] that contains, among other things, classification techniques that are used by TAILOR in both the induction and the hybrid record linkage models. Mainly, two classification techniques are used: induction of decision trees and instance-based learning. The second ready-made tool is called DBGen [15], which is used to generate synthetic data files. The operation of DBGen is controlled by a large number of parameters such as data size, duplication rate, error probabilities in the various fields, etc. Notice that these parameters are instrumental in the generation of controlled studies for comparing the different tools and models included in the system.

4.2 User Interface

TAILOR provides its users with two different ways for interacting with the system. The users can use either a definition language or a graphical user interface. In either way, the user is able to select a searching method, a comparison function, and a decision model, as well as to tune all the required parameters. The values of the parameters determine the functionality of the various components described in Section 4.1. For example, in order for the users to make use of the sorted neighborhood searching method, they should specify values for the two parameters: the sorting key and the window size. Because of space limitations, we are not providing a full description of the interface. However, we refer the interested reader to [8].

5. Experimental Study

This section contains the results of an extensive experimental study that analyzes and empirically compares the various record linkage models and tools have been discussed. The main purpose of this study is to select the best searching method, the best comparison function, and the best decision model, as well as to facilitate the parameter selection process for them.

Section 5.1 compares the string comparison functions discussed in Section 4.1.2. In Section 5.2, experiments are conducted to compare the two searching methods discussed in Section 4.1.1. In Section 5.3, we study the performance of the proposed machine learning record linkage models versus the probabilistic record linkage model.

In our experiments, we exploit synthetic data as well as real data. As mentioned before, a tool called DBGen [15] is used for generating synthetic data. DBGen generates records of people that include the following information for each person: SSN, Name (Last, First, Middle Initial), and Address (Street, City, Zip Code, State). DBGen associates each record with a group number in such a way that records with the same group number represent the same person, i.e., matched records. A Wal-Mart database of 70 Gigabytes, which resides on an NCR Teradata Server running the NCR Teradata Database System, is used for the real data experimental study. The results of this experimental study are reported in Section 5.4.

5.1 Evaluation of String Comparison Functions

Figure 4 shows empirical results for comparing a list of person names using various string comparison functions. The list of names, which is taken from a similar study [26], is known to be the same but misspelled. In order to be able to compare their performance, the comparison values are normalized in the range *[*0,1*]*. The lower the comparison value is, the closer the two strings are to each other. Figure 4 shows that for all the strings, Jaro's algorithm gives the lowest value.

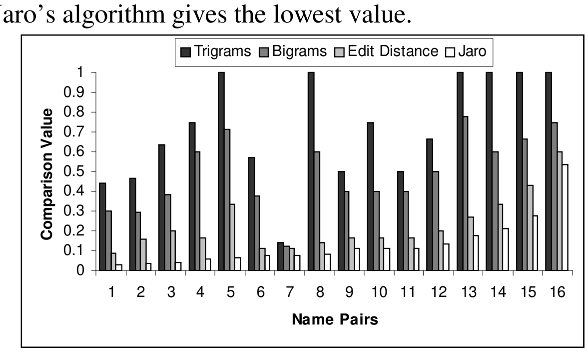
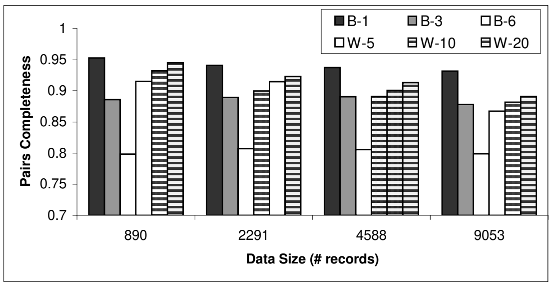


Figure 4. String Comparison Functions

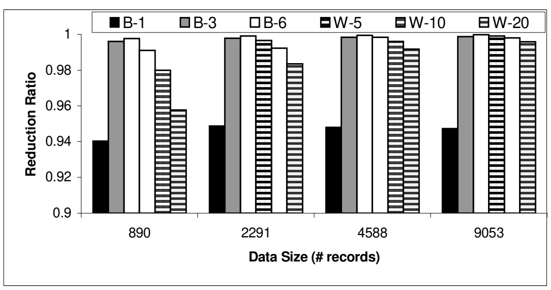
5.2 Comparison of Searching Methods

We use two metrics to compare the effectiveness of the searching methods: the *pairs completeness* metric, and the *reduction ratio* metric. We have conducted two experiments to compare (i) the blocking method for different values of the block key length, and (ii) the sorted neighborhood method for different values of the window size. We use the symbol "B-*x*" to denote the blocking method with block key of length *x*. In addition, we use the symbol "W-y" to denote the sorted neighborhood method of window size y.

Figure 5 shows the results of the first experiment on synthetic data sets of different sizes. The experiment uses the first three characters of the last name combined with the first three characters of the first name as the block key and the sorting key. Figure 5 shows that (i) in the blocking method, the *pairs completeness* value decreases and the *reduction ratio* increases as the value of the block key length increases, and (ii) in the sorted neighborhood method, the *pairs completeness* value increases and the *reduction ratio* decreases as the value of the window size increases.



(a) Pairs Completeness

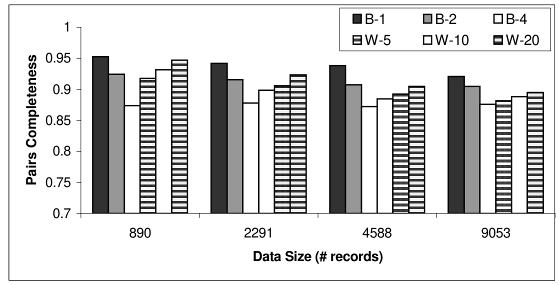


(b) Reduction Ratio

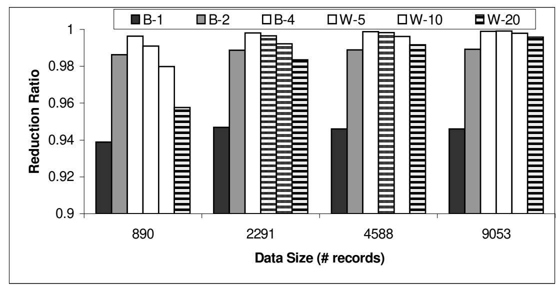
Figure 5. Searching Methods, Experiment 1

Figure 6 shows the results of the second experiment. The Soundex code of the last name is used as the block key and the sorting key. Figure 6 shows a similar tendency in the metrics to the first experiment. However, there is a notable increase in the *pair completeness* metric without a major change in the *reduction ratio*.

A reduced comparison space is better than another if it has a higher reduction ratio and a higher pairs completeness value. Both Figure 5 and Figure 6 show that there is a tradeoff between those two metrics that is similar to the precision recall tradeoff [28]. Similar to the *F score* metric [28] that captures the harmonic mean of precision and recall, we employ a new metric, named *F score*, that is defined as follows: $F\ score = \dfrac{2 \cdot PC \cdot RR}{PC + RR}$. The function ensures that an *F score* will have values within the interval $[0,1]$ with the feature that high values represent better performance than lower values. Figure 7 gives the results using this metric for both the previous experiments. The figure shows that the *F score* values decrease as the data size increases. Figure 7(a) shows that blocking searching method with block key length of *all* (the same length of the sorting key) has the worst performance, and shows that the other methods have approximately the same performance. Figure 7(b) shows that, using Soundex code with large data sizes, blocking with block key length of *all* has a comparable performance.

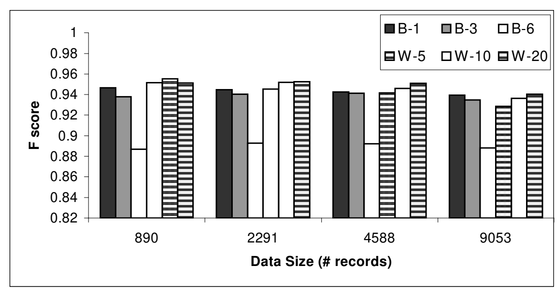


(a) Pairs Completeness

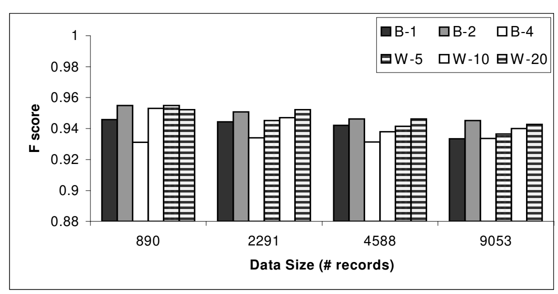


(b) Reduction Ratio

Figure 6. Searching Methods, Experiment 2



(a) Experiment 1



(b) Experiment 2

Figure 7. Searching Methods (F score)

An error that may be encountered in real-world data is the swapping of two field values. For example, a person whose name is "John Smith" may be represented in a record by first name "John" and last name "Smith", and in another record, erroneously, by first name "Smith" and last name "John". TAILOR provides a feature, called *field swapping*, to account for this error. The searching methods are enhanced in order to guarantee that the pair of records with the swapped values is contained in the reduced comparison space. An experiment is performed to evaluate the improvement in performance using this feature. The experiment uses the blocking method where the block key is the Soundex code of the last name. We compare two cases. In the first one, the feature is not set, while in the second, it is set to account for the swapping of the first and the last names. While the *F score* value in the first case is *0.93* on average, in the second case the *F score* value is *0.98* on average. This increase in the *F score* results from an average increase of *0.11* in the pairs completeness of the second case over the first, while the reduction ratio is slightly decreased.

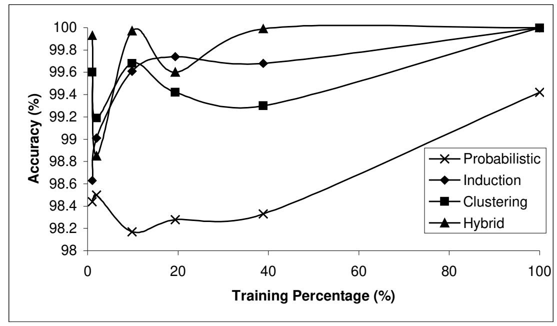
5.3 Comparison of Decision Models

The next experiment compares the various decision models using both *accuracy*, and *completeness* metrics discussed before, and also the percentage of the record pairs that have been predicted as *possibly matched* by the decision model. Figure 8 shows the results of the experiment using a data set of 100,000 records varying the training set size (or varying the reduced comparison space size for the clustering record linkage model).

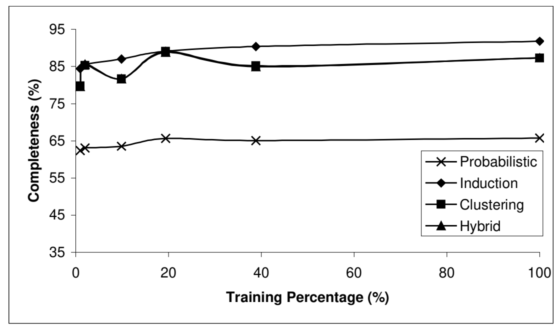
This experiment uses the first three characters of the last name combined with the first three characters of the first name as the sorting key, and uses the sorted neighborhood method with window size of *5* as the searching method. The possible matched set is not defined in the induction record linkage model since the training set does not contain such a label. The figure shows that the machine learning record linkage models outperform the probabilistic record linkage model concerning both the accuracy and the completeness metrics. However, the probabilistic record linkage model has lower percentage of possibly matched record pairs. Therefore, no model proved to be the best under all the metrics, and it totally depends on the user and his criteria to pick the right model for his data.

5.4 Real Data Experimental Study

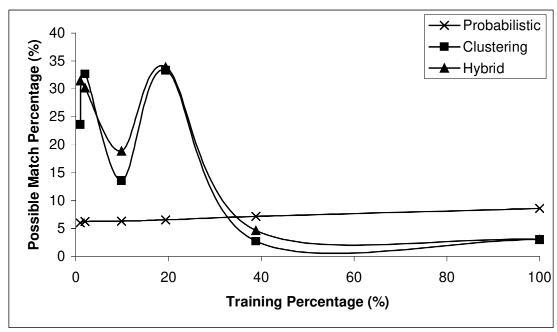
For the experiments with real data, we made use of the *Item* table from the Wal-Mart database. The *Item* table contains half a million items with a total size of 175 Megabytes. The goal of this experiment is to detect duplicate items by applying the developed record linkage models. For example, a specific item such as "TR Orange Juice" may appear in several records, each of which captures a different vendor, a different expiration date, etc.



(a) Accuracy



(b) Completeness



(c) Possible Match Percentage

Figure 8. Decision Models Comparison

An item record contains many fields such as Category Number, Subcategory Number, Primary Description, Secondary Description, Type Code, Color, and Size. The blocking method is used as the searching method where the block key is the Category Number combined with the Subcategory Number. The *field swapping* feature is set to account for the swapping of the Primary Description and the Secondary Description. The size of the reduced comparison space is nearly 200 millions of item pairs, i.e., a reduction ratio of *0.16%*. We use the clustering and the hybrid record linkage models since a training set of item pairs is not available. In the hybrid record linkage model, we apply clustering on *0.1%* of the reduced comparison space, followed by decision tree induction.

Pair	Category Number	Subcategory Number	Primary Description	Secondary Description
1	38	22	ORANGE JUICE	FRESH SQUEEZED 1GAL
	38	22	FRESHLY SQUEEZED	ORANGE JUICE 1 GAL
2	52	13	24/16OZ ARIZONA RASP	RASPBERRY TEA
	52	13	ARIZONA RASPBERRY	ICE TEA 24OZ
3	52	16	24/16 SNAPPLE SWEET	TEA NO/LEMON
	52	16	SNAPPLE LEMON TEA	24-16 OZ BOTTLES
4	52	13	ARIZONA TEA W/LEMON	24-24OZ
	52	13	23.5OZ ARIZONA RASP	24/23.5OZ TEA
5	52	57	ARIZONA RASPBERR TEA	24-16 OZ.
	52	13	24/16OZ ARIZONA RASP	RASPBERRY TEA

Table 3. Wal-Mart Data Item Pairs Examples

The clustering record linkage model predicts *38%* of the item pairs as *matched*, while the hybrid record linkage model predicts *48%* as *matched* item pairs. In order to measure the accuracy of these models, random samples of the item pairs are produced and checked manually. Whereas the accuracy of the clustering record linkage model is found to be *79.8%* on average, the accuracy of the hybrid record linkage model is found to be *76.5%* on average. Measuring the completeness of the models is not feasible since all the item records would have to be checked manually.

Table 3 shows some examples of item pairs. In this table, the first three item pairs are predicted to be matched. Manual review indicates that the third pair is incorrectly predicted as matched. The first item pair demonstrates the importance of setting the *field swapping* feature. The fourth item pair is correctly predicted as unmatched. Although the fifth item pair is a matched item pair, the record linkage model is not able to detect it since this item pair is not included in the reduced comparison space. Notice that the two items have different subcategory numbers. Since the subcategory number is part of the block key, the blocking searching method does not select this item pair in the reduced comparison space.

6. Related Work

Related work falls into two main categories: the record linkage problem and record linkage tools and frameworks.

Hernandez and Stolfo [14] address the record linkage problem under the name *merge/purge*, which is a common name that business organizations use to describe the same problem. The authors propose an equational theory for record linkage. By the term *equational theory*, they mean the specification of inference declarative rules that dictate the logic of record equivalence. Monge and Elkan [23] consider the record linkage problem as an extension of the string matching problem. Their algorithm considers the database record as a string, and it decides the matching status of a record pair based on the distance between the two strings. Dey et al. [7] discuss the same problem under the name *entity matching*, as this name pertains to the system integration and heterogeneous databases areas.

Most recently, record linkage has been investigated in the data cleaning context. Lee et al. [21] extend the equational theory for record linkage to a complete knowledge-based framework for data cleaning. In [4], Caruso et al. demonstrate a data reconciliation tool that is based primarily on a rule-based record linkage model. Galhardas et al. [12] propose a declarative language for the logical specification of data cleaning operations, along with a framework for specifying various data cleaning techniques at the logical and physical level; record linkage is one of these techniques.

7. Conclusions

In this paper, we have presented TAILOR a record linkage toolbox that serves as a framework for the record linkage process. Several in-house developed, as well as public domain tools are bundled into TAILOR. TAILOR is extensible, and hence any proposed searching method, comparison function, decision model, or measurement tool can be easily plugged into the system. We have proposed three machine learning record linkage models that raise the limitations of the existing record linkage models. Our extensive experimental study, using both synthetic and real data, shows that (i) the machine learning record linkage models outperform the probabilistic record linkage model with respect to the accuracy and the completeness metrics, (ii) the probabilistic record linkage model identifies a lesser percentage of possibly matched record pairs, (iii) both the clustering and the hybrid record linkage models are very useful, especially in the case of real applications where training sets are not available or are very expensive to obtain, and (iv) Jaro's algorithm performs better than the other comparison functions.

References

[1] D.W. Aha. Tolerating Noisy, Irrelevant and Novel Attributes in Instance Based Learning Algorithms. *International*

Journal of Man-Machine Studies, 36(1), pages 267-287, 1992.

[2] D. Bitton and D. DeWitt. Duplicate Record Elimination in Large Data Files. *ACM Trans. on Database Systems*, 8(2), pages 255-265, June 1983.

[3] P.S. Bradley and U.M. Fayyad. Refining Initial Points for K-Means Clustering. In *Proc. of the 15th Int. Conf. on Machine Learning*, pages 91-99, San Francisco, CA, 1998.

[4] F. Caruso, M. Cochinwala, U. Ganapathy, G. Lalk, and P. Missier. Telcordia's Database Reconciliation and Data Quality Analysis Tool. In *Proc. of the 26th VLDB Int. Conf. on Very Large Databases*, Cairo, Egypt, September 2000.

[5] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha. Efficient Data Reconciliation. Technical Report, Telcordia Technologies, February 1998.

[6] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39, pages 1-38, 1977.

[7] D. Dey, S. Sarkar, and P. De. A Probabilistic Decision Model for Entity Matching in Heterogeneous Databases. *Management Sciences*, 28(1), pages 31-46, 1998.

[8] M.G. Elfeky, V.S. Verykios, and A.K. Elmagarmid. The TAILORView Graphical User Interface. Technical Report, Department of Computer Sciences, Purdue University, Summer 2001.

[9] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases (a survey). *AI Magazine*, 17(3), pages 37-54, 1996.

[10] I.P. Fellegi and A.B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64, pages 1183-1210, 1969.

[11] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*, Prentice-Hall, 1992.

[12] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In *Proc. of the 27th VLDB Int. Conf. on Very Large Databases*, Roma, Italy, September 2001.

[13] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate String Joins in a Database (Almost) for Free. In *Proc. of the 27th VLDB Int. Conf. on Very Large Databases*, Roma, Italy, September 2001.

[14] M.A. Hernandez and S.J. Stolfo. The Merge/Purge Problem for Large Databases. In *Proc. of the 1995 ACM SIGMOD Int. Conf. on Management of Data*, San Jose, California, May 1995.

[15] M.A. Hernandez and S.J. Stolfo. Real World Data is Dirty: Data Cleansing and the Merge/Purge Problem. *Journal of Data Mining and Knowledge Discovery*, 2(1), pages 9-37, 1998.

[16] J.A. Hylton. Identifying and Merging Related Bibliographic Records. Master's Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1996.

[17] M.A. Jaro. UNIMATCH: A Record Linkage System, User's Manual. Washington DC: U.S. Bureau of the Census, 1978.

[18] M.A. Jaro. Advances in Record Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406), pages 414-420, 1989.

[19] D. Knuth. *The Art of Computing Programming, Volume III*. Addisson-Wesley 1973.

[20] R. Kohavi, D. Sommerfield, and J. Dougherty. Data Mining using MLC++: A Machine Learning Library in C++. *IEEE Tools with Artificial Intelligence*, pages 234-245, 1996. http://www.sgi.com/tech/mlc/index.html.

[21] M.L. Lee, T.W. Ling, and W.L. Low. IntelliClean: A Knowledge-Based Intelligent Data Cleaner. In *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, MA, August 2000.

[22] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[23] A.E. Monge and C.P. Elkan. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In *Proc. of DMKD'97 the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Tucson, Arizona, May 1997.

[24] H.B. Newcombe. Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family histories. *American Journal of Human Genetics*, 19(3), 1967.

[25] H.B. Newcombe, J.M. Kennedy, S.J. Axford, and A.P. James. Automatic Linkage of Vital Records. *Science*, 130, pages 954-959, 1959.

[26] E.H. Porter and W.E. Winkler. Approximate String Comparison and its Effect on an Advanced Record Linkage System. U.S. Bureau of the Census, Research Report, 1997. http://www.census.gov/srd/papers/pdf/rr97-2.pdf.

[27] J.R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1, pages 81-106, 1986.

[28] W. Shaw, R. Burgin, and P. Howell. Performance Standards and Evaluations in IR Test Collections: Vector-Space and Other Retrieval Models. *Information Processing and Management*, 33(1), pages 15–36, 1997.

[29] T.F. Smith and M.S. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147, pages 195-197, 1981.

[30] V.S. Verykios. A Decision Model for Cost Optimal Record Matching. In *National Institute of Statistical Sciences Affiliates Workshop on Data Quality*, Morristown, New Jersey, 2000.

[31] V.S. Verykios, M.G. Elfeky, A.K. Elmagarmid, M. Cochinwala, and S. Dalal. On The Accuracy and Completeness of The Record Matching Process. In *Proc. of the 2000 Conf. on Information Quality*, Boston, Massachusetts, October 2000.

[32] V.S. Verykios, A.K. Elmagarmid, and E.N. Houstis. Automating the Approximate Record Matching Process. *Journal of Information Sciences*, 126(1-4), pages 83-98, July 2000.

[33] W.E. Winkler. Using the EM Algorithm for Weight Computation in the Fellegi-Sunter Model of Record Linkage. In *Proc. of the Section on Survey Research Methods*, American Statistical Association, pages 667-671, 1988.

[34] W.E. Winkler. The State of Record Linkage and Current Research Problems. U.S. Bureau of the Census, Research Report, 1999. http://www.census.gov/srd/papers/pdf/rr99-04.pdf.

[35] J.J. Zhu and L.H. Ungar. String Edit Analysis for Merging Databases. In *KDD 2000 Workshop on Text Mining*, Boston, Massachusetts, August 2000.