Evaluation of Record Linkage Methods for Iterative Insertions

M. Sariyar; A. Borg; K. Pommerening
Institute of Medical Biostatistics, Epidemiology and Informatics, University Medical Center, Mainz, Germany

Keywords
Record linkage, object identification, decision trees, support vector machines, EM algorithm

Summary
Objectives: There have been many developments and applications of mathematical methods in the context of record linkage as one area of interdisciplinary research efforts. However, comparative evaluations of record linkage methods are still underrepresented. In this paper improvements of the Fellegi-Sunter model are compared with other elaborated classification methods in order to direct further research endeavors to the most promising methodologies.
Methods: The task of linking records can be viewed as a special form of object identification. We consider several non-stochastic methods and procedures for the record linkage task in addition to the Fellegi-Sunter model and perform an empirical evaluation on artificial and real data in the context of iterative insertions. This evaluation provides a deeper insight into empirical similarities and differences between different modelling frames of the record linkage problem. In addition, the effects of using string comparators on the performance of different matching algorithms are evaluated.
Results: Our central results show that stochastic record linkage based on the principle of the EM algorithm exhibits best classification results when calibrating data are structurally different to validation data. Bagging, boosting together with support vector machines are best classification methods when calibrating and validation data have no major structural differences.
Conclusions: The most promising methodologies for record linkage in environments similar to the one considered in this paper seem to be stochastic ones.

Correspondence to:
Murat Sariyar
University Medical Center
Obere Zahlbacher Str. 69
55131 Mainz
Germany
E-Mail: sariyar@imbei.uni.mainz.de

Methods Inf Med 2009; 48: 429–437
doi: 10.3414/ME9238
prepublished: August 20, 2009

1. Introduction

The task of linking records concerning personal data is more and more relevant in times of widespread and distributed registration of such information[a].

The main goal of record linkage procedures is the minimization of synonym and homonym errors when merging data into one main target file. Synonym errors occur when data concerning one person are stored in at least two records of the target file. Homonym errors, which are harder to find than synonym errors, are made when data of two or more different persons are classified as equal. Usually record linkage means linking two files together; in this paper, we consider the case of iterative insertions of data into a database. The fundamental difference between iterative insertions and the canonical context of record linkage procedures (merging two or more files into one file) lies in the importance and timeframe of a representative model. Whereas a model which will be applied for some time has to be calibrated for iterative insertions, for the canonical context there is no need of a model to be representative for a long time because of the uniqueness of the linking process.

The classical record linkage problem starts with two sets of objects A and B, stemming from a source population Ω[b]. Objects y_i, $i = 1, ..., N$, from Ω, which constitute sets A and B, are mapped into an L-dimensional feature space $C = C_1 \times \Lambda \times C_L$ through data generation processes $\alpha: A \rightarrow C$ and $\beta: B \rightarrow C$, so that the objects in A and B are represented by data files D_A and D_B with rows of format $(c_1, ..., c_L)$. On the basis of these feature values, one has to decide which tuples of the Cartesian product $D_A \times D_B$ or of some of its subsets represent one object, and which ones are composed of data from different objects. Those tuples of the set $A \times B$ which correspond to one object are called matches (M). The complementary tuples to M in $A \times B$ are called non-matches (U).

A comparison function $f: C \times C \rightarrow \Gamma$ with $\dim(\Gamma) = k \leq L$ is used to create comparison patterns $\gamma = (\gamma_1, ..., \gamma_k)$ for every tuple in $D_A \times D_B$. Based on these comparison patterns the decision concerning the matching status of the tuples is made. Frequently used values for Γ are $\{0, 1\}^k$, where 1 corresponds to the comparison value "equal" and 0 stands for "unequal". After these necessary preparations, a modeling frame for evaluating comparison patterns has to be chosen.

The classical record linkage framework is based on probabilistic models and was formally justified by Fellegi and Sunter [1]. The outset of the probabilistic modeling framework is constituted by existence assumptions concerning the conditional probabilities $P(\gamma \mid M)$, $P(\gamma \mid U)$, $P(\gamma_i \mid M)$, $P(\gamma_i \mid U)$, $i = 1, ..., k$. In this context the functions (α, β) represent some error inducing stochastic data generation mechanism, so that γ, defined through $\gamma = f \circ (\alpha, \beta): A \times B \rightarrow \Gamma$, forms a ran-

[a] Issues such as data privacy or data security will not be discussed here.

[b] The introduction of multisets will not be considered.

Downloaded by: Karolinska Institutet. Copyrighted material.

dom variable. When using binary comparison patterns likelihood ratios are used to determine the matching-status of object pairs:

$$\frac{P(\gamma_i \mid M)}{P(\gamma_i \mid U)}, \text{ for all } i.$$

Probability $P(\gamma_i = 1 \mid M)$ must be greater than probability $P(\gamma_i = 1 \mid U)$, otherwise the probability of a random agreement of one feature would be greater than agreement in case of a match. For computational convenience, the logarithm of the likelihood ratio is used for performing record linkage:

$$w_\gamma = \log\left(\frac{P(\gamma \mid M)}{P(\gamma \mid U)}\right)$$

This log likelihood is called *global weight*. For the global weight, either one or two thresholds have to be determined in order to classify the underlying comparison vector. In this paper thresholds will be determined using training data. Where one threshold is determined, every vector will be classified as a match if its weight is greater than this threshold, otherwise it will be classified as a non-match. Where there are two thresholds, a range for non-determination between these two thresholds is allowed, necessitating clerical review for the determination of the definite class of comparison vectors. If a global weight falls below the lower threshold, the underlying two objects are classified as non-equal, and if it exceeds the higher threshold, the underlying two objects are classified as equal.

Record linkage needs some preliminary steps. First, the structure of the data sets has to be determined and, if necessary, the data have to be transformed. This involves the determination of relevant features and their domains. A stochastic model for the conditional probabilities $P(\cdot \mid M)$ and $P(\cdot \mid U)$ is then assumed. The last preliminary step consists of parameter estimations concerning models for $P(\cdot \mid M)$ and $P(\cdot \mid U)$. A major simplifying hypothesis in many cases is the so-called conditional independence assumption, which comes down to:

$$P(\gamma \mid M) = \prod_{i=1}^{k} P(\gamma_i \mid M)$$

and

$$P(\gamma \mid U) = \prod_{i=1}^{k} P(\gamma_i \mid U)$$

This hypothesis means that all components of comparison patterns take values independently of each other under the condition that γ stems from M or U. In reality, this assumption of conditional independence is violated in most cases. For instance, sex depends on the first name, so the corresponding parts of comparison patterns are dependent of each other. However, the conditional independence assumption still generates good results for record linkage procedures. This could be due to the fact that overall errors and not particular errors are considered. These superimpositions of errors are well captured by the conditional independence hypothesis. If dependence should be explicitly considered, application of log-linear hierarchical models is suitable[c].

The expectation-maximization (EM) algorithm, introduced by Dempster et al. [3], is the most important estimation method for latent class models. In the case of record linkage the latent class variable reveals the status of the object pairs as match, non-match and casually as indefinite. The EM-algorithm facilitates relaxations of the conditional independence assumption in order to operate with more realistic dependency structures within probabilistic models[d].

The purpose of reducing computational burden when deploying a record linkage procedure for linking large files is relevant. To achieve this blocking is applied. One or more features are selected as grouping variables and only pairs with agreement in these blocking variables are considered. In order to take account of errors in grouping variables, blocking should be run several times with different variables.

String metrics are frequently used in record linkage applications. This entails weighting global weights according to the similarity of feature characteristics, thereby enhancing the differentiation between comparison patterns of data pairs.

2. Methods

2.1 Extensions for the Fellegi-Sunter Model

2.1.1 EM Algorithm

EM algorithms are usually applied when parameterized densities of a probability distribution are known. In this case, they are only extended maximum likelihood procedures with computation of the expected value over the unknown class variable as an intermediary step. However, the concept is so general that it adapts to a wider class of probability structures. When a discrete probability structure which can be modeled as probabilities of contingency table cells is available, latent variables correlate with collapsing cells. The size of a full contingency table is the product of the number of latent classes and the size of the observable contingency table.

The goal is to estimate the full contingency table based on some parameterized categorical probability distribution and on the observable contingency table.

An algorithm for computing such a full contingency table with cell frequencies e is based on fitting general log-linear models (*gllm*). The denotation gllm stems from the fact that for the logarithm of the expected value $\log \mathrm{E}[e]$ a linear model $M\theta$ is assumed (M is a known $K \times L$ design matrix with non-negative components m_{ij}, which sum to one for each row: $\sum m_{ij} = 1$; θ is an unknown $L \times 1$ parameter vector)[e]. Computing expected values in this context means estimating the full contingency table. Espeland and Odoroff [8] show that estimation of the expected values can be performed as an iterative proportional adjustment with values stemming from preceding maximization steps. The maximization step is based on the generalized iterative scaling method of Darroch and Ratcliff [9], which is equivalent to the maximization of the entropy of the searched discrete probability distribution. This EM algorithm is still not considered in the literature of record linkage, although it needs no assumption concerning the dependence structure of available attributes and is suitable for the record linkage context.

[c] A good introduction for log-linear models is [2].

[d] Cf. for example [4] and [5].

[e] Cf. for further information [6] and [7].

Downloaded by: Karolinska Institutet. Copyrighted material.

© Schattauer 2009

2.1.2 Application of String Metrics

In [10] the record linkage problem is regarded as extension of a string identification task when errors occur. In this perspective the usage of string metrics is of crucial importance. In contrast to phonetic codes, string metrics deal with errors in a more sophisticated way[f]. Results of string metrics are values between zero and one, reflecting similarity between strings compared. They are used to adjust the corresponding individual weights for exact agreement in the Fellegi-Sunter model.

Important string metric algorithms are N-grams, edit-distance and string metric procedures of Jaro and Winkler. Our internal empirical studies have shown that differences of the mentioned string metrics are negligible. Yancey [11] established similar results.

In this paper the string metric established by Jaro [12] and extended by Winkler [13] is used. Their approach is based on the computation of related letters and transpositions. If A is the number of letters of string $v = \{v_1, ..., v_A\}$ and B is the number of letters of string $w = \{w_1, ..., w_B\}$, then letters v_i and w_j are defined as related when the following holds: $v_i = w_j$ and $|i - j| < \lfloor (0.5 \cdot \max(A, B) - 1) \rfloor$.

Only one-to-one relations between letters are allowed. Transpositions are defined as related letters with $i \neq j$. The most important similarity measure is

$$S_{JW} = \frac{1}{3}\left(\frac{Z}{B} + \frac{Z}{A} + \frac{Z-T}{Z}\right),$$

where Z is the number of related letters and T is the number of transpositions.

2.2 Further Models

Generalization and abstraction of the problem of record linkage as object identification broaden the spectrum for further modeling frameworks and solutions. There is no need for a stochastic modeling of the linkage task when sufficient data is available. In such cases the structure of the data can be used and exploited in such a manner that deterministic models for object identification are adaptable. Here we will consider decision tree procedures with the two extensions *boosting* and *bagging*, and *support vector machines* as elaborated methods with supervised learning for non-stochastic record linkage.

2.2.1 Decision Tree Models

Decision tree models represent relevant information and structure of available data in the form of binary source trees. At the outset, the root of such a tree contains all training samples. The root node is split into two child nodes according to some purity measure which quantifies the clustering of objects with equal values for the class membership variable based on this split. The values of every feature are examined and the split is performed according to that feature value which maximizes the purity.

This process is continued recursively until changes of the impurity measure reach a lower bound or the number of objects in a node is lower than or equal to a fixed minimum size. An edge of a decision tree represents the value of a feature variable for which the objects in the parent node are allocated to the child node of that edge. The leaves represent those classes to which the objects in these leaves are assigned. The nearer nodes are to the root the more important the features assigned to them are. The goal is to find a surjective mapping from the classes to the leaves of a decision tree that is as small as possible.

The most important algorithm for creating non-aggregated decision trees is CART (Classification and Regression Trees), developed in [14]. The algorithm has two steps. First a maximal decision tree with a very simple stopping rule is generated and then this tree is pruned in order to avoid overfitting.

In order to improve performance and stabilize decision trees, which can differ substantially even for slight changes of the training data, weighted aggregations of decision trees are considered. Two important ensemble methods are *boosting* and *bagging*. *Bagging* (bootstrap aggregating) generates bootstrap samples from the whole training data and then aggregates decisions of the generated decision trees via majority vote. For more information we refer to [15]. *Boosting* has a different approach of aggregating decision trees compared to *bagging*. Instead of generating bootstrap samples, the same training data is used to weight wrongly classified data higher in order to get different characteristics in the training data for generating different decision trees. The most important boosting algorithm for decision trees is AdaBoost, which is used in this paper. For details about AdaBoost we refer to [16].

2.2.2 Support Vector Machines

The denotation *support vector machines* stands for classification methods which originated in the field of machine learning. These methods are destined for performing classification through determination of separating vectors with maximum distance of each other in some feature space[g]. The margin maximization on some training data is performed in order to get the most promising way of differentiating evaluation data with similar structural properties as the training data. A linear separation of training data according to their classes is often impossible and so two extensions which are mostly combined exist. The first extension pertains to a deployment of slack variables for the linear hyperplanes defining inequalities which were injected in the distances computing objective function with multiplied constant penalty terms. A second extension concerns mapping feature vectors x in the origin space to a higher-dimensional space according to a function ϕ in order to enlarge the possibility of linear separations in the new space. This second extension can be simplified because in the dual form the transformed vectors emerge within inner products. Instead of mapping the original vectors through a function to a larger space one can search for a so-called kernel function corresponding to the inner products. The finding of an appropriate kernel is in fact a trial-and-error work. A frequently used kernel is the so-called radial basis function K_G, which is defined as:

$$K_G(x, y) := \exp(-\sigma \| x - y \|^2), \sigma \in \mathbb{R}.$$

[f] In our pre-tests string metrics were far superior to phonetic codes. It's important to emphasize that using phonetic codes in addition to the underlying names would violate independence requirements.

[g] For further background and introduction, we refer to [17] and [18].

© Schattauer 2009

Downloaded by: Karolinska Institutet. Copyrighted material.

Nr.	Comparison name	Description
1	match_fname_c1	Comparison of first names (first component)
2	match_fname_c2	Comparison of first names (second component)
3	match_lname_c1	Comparison of names (first component)
4	match_lname_c2	Comparison of names (second component)
5.1	match_dob	Comparison of birthday en bloc
5.2.1	match_by	Comparison of day of birth
5.2.2	match_bm	Comparison of month of birth
5.2.3	match_bd	Comparison of year of birth

Table 1 Denotation of feature comparisons

2.2.3 Application of Object Identification Methods for Record Linkage

A crucial model assumption for methods relying solely on the structure of given data is representativity of the training data. This assumption can only be made plausible but is not provable. In this respect, it is comparable with the model assumption of some random processes for stochastic procedures. In both cases, not only the characteristics of available data are relevant, but also those of the whole data of the context. The training data should therefore be chosen randomly in order to reinforce the assumption of representativity.

Unlike in procedures of stochastic record linkage, the attributes within the considered methods of object identification cannot be quantitatively evaluated with respect to their contribution to correct classifications. In the context of decision trees, at least the relevance of attributes is identifiable as the depth of their appearance in a tree: the lesser the deepness, the higher the relevance of an attribute. The advantage of hierarchical structuring of information disappears when decision trees are aggregated. In general, it can be said that elaborated methods and those aggregating simpler ones most often do not exhibit relevant structures of the data involved.

3. Empirical Evaluation

3.1 Implementations

3.1.1 Data Structure

The structure of data stems from the context of a cancer registry. They represent individual data common for record linkage in registries and medical research networks[h]. The corresponding matching attributes are listed in ▶Table 1. Sex as a sparse information-revealing attribute should only be used as a blocking variable. The attribute date of birth is included twice, which produces two cases in order to evaluate the effect of aggregating information:

- dz: comparison with aggregated date of birth (match_dob) in *yyyymmdd*-format
- dg: comparison with separated date of birth as year (*yyyy*), month (*mm*) and day (*dd*)

Further, we consider binary comparisons (equal = t, not equal = f) and comparisons based on the Jaro-Winkler string metric S_{JW}. The combination of these two distinctions leads to following comparison cases:

- (binary, dg): binary comparisons with separated birthday,
- (fuzzy, dg): using Jaro-Winkler string metric with separated birthday,
- (binary, dz): binary comparisons with birthday en bloc,
- (fuzzy, dz): using Jaro-Winkler string metric with birthday en bloc.

3.1.2 Methods

For the methods considered, established parameter values are chosen since we strive for generalizable applications and not for classification error minimization on a specific data appearance. All algorithms except *FEBRL* are implemented in the statistical software environment *R*.

As *stochastic methods* based on the principles of the Fellegi-Sunter model we evaluate the following:

FEBRL (Freely Extensible Biomedical Record Linkage) is an implementation of a simple algorithm for the Fellegi-Sunter model, written in the scripting language Python and described in [20]. It is based on the assumption of conditional independence. Necessary conditional probabilities have to be set manually. As input, non-paired data are injected and pairing is done as part of the procedure. In order to allow comparisons with other algorithms, an artificial blocking variable is introduced to produce the desired pairs of data objects.

Epilink [21] is a simple procedure within the scope of the Fellegi-Sunter model and actually designed for using a string metric $s(\cdot\,,\cdot)$. A global weight S_{ij} for some data pair $(\alpha(a_i),(\beta(b_j)) = (\alpha_i,\beta_j) = ((\alpha_{i1},...,\alpha_{in}), (\beta_{i1},...,\beta_{in}))$ is computed as:

$$S_{ij} = \frac{\sum_k w_k s(\alpha_{ik},\beta_{jk})}{\sum_l w_l}$$

where the w_j is the individual weight for comparison attribute j and is constituted by error rate e_j and average frequency f_j of values of comparison attribute j:

$$w_j = \log_2 \frac{1-e_j}{f_j}$$

Average frequency f_j has to be estimated using available data.

EM (using the EM algorithm): the implementation of the algorithm introduced in Section 2.1 in *R* is based on [22] and operates with absolute frequencies. Among other things, it generates a full contingency table output. For every binary comparison pattern $\tilde{\gamma}$, the counts for links $N_M(\tilde{\gamma})$ and non-links $N_U(\tilde{\gamma})$ have to be transformed into probability estimations:

$$\hat{P}(\tilde{\gamma} \mid M) = \frac{N_M(\tilde{\gamma})}{\sum_{\tilde{\gamma}} N_M(\tilde{\gamma})}$$

[h] For further details see [19].

© Schattauer 2009

Downloaded by: Karolinska Institutet. Copyrighted material.

and

$$\hat{P}(\tilde{\gamma} \mid U) = \frac{N_U(\tilde{\gamma})}{\sum_{\tilde{\gamma}} N_U(\tilde{\gamma})}$$

Consideration of string metrics in comparison patterns is accomplished through generation of binary patterns from continuous comparison patterns using thresholds. Concretely, if a value of a string metric s_j^i (Jaro-Winkler) for comparison attribute j in data pair i exceeds 0.9, then it will be transformed to one, else to zero. The corresponding weight $\tilde{w}_i$ for this data pair is shown in ▶Figure 1. The string metric considers errors in duplicates, which explains this formula. If w_i is the weight without consideration of string metrics then

$$\tilde{w}_i = w_i + \log\left(\prod_{j:s_j^i \geq 0.9} s_j^i\right)$$

We have slightly modified the algorithm in [22] in order to allow initial values for expected frequencies as externally-given input.

For each stochastic method considered, computation of thresholds is performed through minimization of classification error on training data.

The following *non-stochastic* methods are considered in the subsequent evaluation: *CART* (decision trees): we will use the implementation of CART by Thernau in R [23], which is available as package *rpart*. The separation of induction of a maximal decision tree and pruning generates the advantage of plotting the relative complexity parameters against cross validation errors in order to choose an optimal complexity parameter value. The Gini index is the preset objective function and will be used here as well. Once a decision tree is determined, a new classification is performed through the generic R-function *predict*.

Bagging: it is implemented in R within the scope of the package *ipred* and performs bootstrap aggregating in a straightforward manner. As default, a value of 50 is set as the number of bootstrap samples. For further details, we refer to [24].

Boosting: it is implemented in *R* in package *ada* and needs routines of *rpart*. For invoking AdaBoost, option type=discrete must be set.

Fig. 1

$$\tilde{w}_i = \log\left(\prod_j \frac{m_j^{\delta_j^i}(1-m_j)^{(1-\delta_j^i)}(s_j^i)^{\delta_j^i}}{u_j^{\delta_j^i}(1-u_j)^{(1-\delta_j^i)}}\right), \begin{cases}\delta_j^i = 1, & s_j^i \geq 0.9\\ \delta_j^i = 0, & s_j^i < 0.9\end{cases}$$

SVM (Support Vector Machines): *R*-package *e1071* contains an interface to the library *libsvm*, which has its foundation in [25] and [26] as outlined in Section 3.2.

3.2 Results on Artificial Data

As data pool, we use the characteristics of approximately 48,000 single data items of a German cancer registry. The data generation module of FEBRL is used to generate data from these characteristics. Data pairs are constructed and transformed into binary comparison patterns, or patterns with string metric values as components. We fix the number of non-matches as 100,000 and add matches in proportion of 1:5 (*K5*) and 1:20 (*K20*) in order to evaluate the effect of different match/non-match ratios. Alpha errors, beta errors, as well as the accuracy values are computed. If $|V_{x,y}|$ represents the amount of classification decisions x when class y is the real class and classes m for a match and u for a non-match exist, then the measures are defined as:

α-error: $$\frac{|V_{u,m}|}{|V_{m,m}| + |V_{u,m}|}$$

β-error: $$\frac{|V_{m,u}|}{|V_{u,u}| + |V_{m,u}|}$$

accuracy: $$\frac{|V_{m,m}| + |V_{u,u}|}{|V_{m,u}| + |V_{m,m}| + |V_{u,m}| + |V_{u,u}|}$$

▶Result tables are shown on the following pages. Only four decimal places are considered. However, for computations of minimal errors and maximal accuracy values, which are tagged as bold, a higher exactness is considered.

3.2.1 Comments

- The lowest rates of α-errors for all methods considered and independent of duplicate ratios are found when using string metrics. With the exception of pruned decision trees, comparison with birthday en bloc does not yield any substantial difference to comparison with separated birthday components, both for binary and fuzzy comparison. In binary cases, the lowest α-errors are found for dg. The comparison case (binary, dg) yields only slightly higher α-errors than in fuzzy cases.
- With the exception of higher β-errors for decision trees for all comparison cases, β-errors are similar and on a very low level. β-errors are generally lower than α-errors due to the relative high number of non-duplicates.
- The values of the accuracy measure depend largely on the α-errors, hence using string metrics yields the best results for all comparison cases and independent of the duplicate ratio. Further comments on the α-errors hold here as well[i].
- Considered different duplicate ratios differ only on irrelevant levels. The methods form three distinct groups for which evaluation measures are quite similar:
  - *Bagging*, *boosting* and *SVM*: they exhibit lowest α-errors and highest accuracy values. When using string metrics they also show lowest β-errors. Methods with weight computation show lower β-errors solely for binary comparisons.
  - Methods with weight computation: (FEBRL, Epilink, EM): this group produces the second best results for α-errors and accuracy values. Regarding β-errors these methods exhibit the best results together with the first

[i] One minor difference between K20 and K5 occurs in (fuzzy, dz), where K20 produces slightly higher values for the accuracy measure because of the reduced duplicate ratio.

© Schattauer 2009

Downloaded by: Karolinska Institutet. Copyrighted material.

alpha	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.0100	0.0034	0.0102	0.0034
prune	0.0075	0.0034	0.2709	0.0034
bagg	0.0026	0.0002	0.0094	0.0001
boost	0.0029	**0.0001**	0.0096	**0.0001**
em	0.0055	0.0033	0.0332	0.0058
epilink	0.0048	0.0021	0.0145	0.0092
febrl	0.0094	0.0007	0.0145	0.0034
libsvm	0.0025	0.0001	0.0093	0.0001

Table 2
Alpha errors for K5

beta	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.0005	0.0017	0.0004	0.0017
prune	0.0033	0.0017	0.0004	0.0017
bagg	0.0002	0.0000	0.0005	0.0000
boost	0.0001	0.0000	0.0004	0.0000
em	0.0001	0.0000	**0.0000**	0.0000
epilink	0.0001	0.0000	0.0004	**0.0000**
febrl	0.0001	0.0000	0.0004	0.0001
libsvm	0.0002	0.0000	0.0005	0.0000

Table 3
Beta errors for K5

acc.	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.9979	0.9981	0.9980	0.9980
prune	0.9960	0.9981	0.9545	0.9980
bagg	0.9994	0.9999	0.9981	1.0000
boost	0.9994	1.0000	0.9980	**1.0000**
em	0.9990	0.9994	0.9945	0.9990
epilink	0.9991	0.9996	0.9973	0.9985
febrl	0.9984	0.9999	0.9973	0.9994
libsvm	0.9995	1.0000	0.9981	1.0000

Table 4
Accuracies for K5

alpha	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.0116	0.0082	0.0124	0.0088
prune	0.0078	0.0080	0.2650	0.0084
bagg	0.0026	0.0006	0.0114	**0.0002**
boost	0.0026	0.0006	0.0124	0.0004
em	0.0258	0.0154	0.0386	0.0174
epilink	0.0178	0.0118	0.0276	0.0118
febrl	0.0168	0.0032	0.0274	0.0032
libsvm	0.0022	0.0004	0.0114	**0.0002**

Table 5
Alpha errors for K20

group. Altogether, the method for obtaining weights seems to have no impact on the results.

– Simple decision trees: they show the worst results. Pruned decision trees produce even poorer results for binary comparison patterns than non-pruned decision trees.

In summary, the non-stochastic elaborated methods considered can extract enough information from available artificial data in order to produce best classification results. Methods with weight computation are only preferable when β-errors are of major concern.

3.3 Results on Real Data

In this section our aim is to evaluate how real data are classified by our methods when they are calibrated with artificial data. It is obvious that only the cases where artificial data are structurally different from the real data are important. Data from K5 are used as calibrating data. In real data, 5492 duplicates exist to which non-duplicates are added according to the desired proportion of duplicates to non-duplicates. Two cases are evaluated:

- calibrating methods with artificial data from K5 (n = 120,000) and validation with real data (n = 108,580) with duplicate ratio of 1:20 (R20) ⇒ (K5, R20)
- calibrating methods with artificial data from K5 (n = 120,000) and validation with real data (n = 82,981), whose non-duplicates have shown string metric values (Jaro-Winkler measure) ≥0.8 for the first components of first name and name. ⇒ (K5, SIM).

3.3.1 Comments

- On artificial data *bagging*, *boosting* and *SVM* exhibit the best classification results. This is no longer the case when using real data. This shows that these methods need structural similarity of training and validation data in order to produce excellent results.
- Using the Jaro-Winkler string metric on real data no longer produces the best results in contrast to evaluation results on artificial data. Binary comparison patterns

© Schattauer 2009

Downloaded by: Karolinska Institutet. Copyrighted material.

lead to better results because of their generalizability.

- α-errors are no longer the major error source. This reversion makes the accuracy measure predominantly dependent on the β-error which emphasizes the importance of low β-errors for generalizability. The EM method produces the lowest β-errors in all cases.
- There is no longer a grouping of methods regarding evaluation measures. Results are more irregular. Only the overall best and worst classification results of EM and simple decision trees are stable. This shows that the method for obtaining weights is crucial when calibration and validation have different structures.
- On artificial data pruning of decision trees generally impairs classification results of decision trees. On (K5, R20) for (binary, dg) pruning enhances classification results in comparison with non-pruned decision trees. The desired generalizability of pruning is attained, but only for the deepest trees on our data.
- Again, no relevant differences or trends can be detected for cases with dg and dz. Decision trees suggest use of separated birthday components in order to exploit all of the information.

In summary, we advise that in cases where uncertainty regarding the representativity of training data exists, the EM method should be preferred. Otherwise *bagging*, *boosting* or *SVM* should be implemented. Binary comparison cases produce only slightly worse classification errors than fuzzy cases for artificial data, and comparison with separated birthday components yields the best exploitation of information. Confinement on (binary, dg) is thus advised. This saves resources which would otherwise be used for the computation of string metric values.

4. Discussion and Conclusion

It becomes apparent that the non-stochastic methods considered in this paper are only preferable when training data and validation data have the same structure. They rely too much on the assumption of representativity of training data. An assumption concerning

Table 6 Beta errors for K20

beta	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.0004	0.0000	0.0004	0.0000
prune	0.0036	0.0014	0.0003	0.0016
bagg	0.0001	0.0000	0.0005	0.0000
boost	0.0001	0.0000	0.0004	0.0000
em	0.0000	0.0000	**0.0000**	0.0000
epilink	0.0000	0.0000	**0.0000**	**0.0000**
febrl	0.0000	0.0000	**0.0000**	0.0000
libsvm	0.0001	**0.0000**	0.0005	**0.0000**

Table 7 Accuracies for K20

acc.	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.9991	0.9996	0.9990	0.9996
prune	0.9962	0.9983	0.9871	0.9981
bagg	0.9998	1.0000	0.9990	1.0000
boost	0.9998	1.0000	0.9990	1.0000
em	0.9987	0.9993	0.9982	0.9992
epilink	0.9991	0.9994	0.9987	0.9994
febrl	0.9992	0.9998	0.9987	0.9998
libsvm	0.9998	1.0000	0.9990	**1.0000**

Table 8 Alpha errors for (K5, R20)

alpha	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.0024	0.0078	0.0002	0.0080
prune	0.0024	0.0078	0.0348	0.0080
bagg	**0.0002**	0.0004	**0.0002**	0.0007
boost	**0.0002**	0.0040	**0.0002**	0.0004
em	0.0035	0.0009	0.0035	0.0005
epilink	**0.0002**	0.0005	**0.0002**	0.0011
febrl	0.0024	0.0064	0.0024	0.0064
libsvm	**0.0002**	0.0027	**0.0002**	0.0027

Table 9 Beta errors for (K5, R20)

beta	(binary, dg)	(fuzzy, dg)	(binary, dz)	(fuzzy, dz)
rpart	0.0006	0.0011	0.0004	0.0011
prune	0.0039	0.0011	0.0003	0.0011
bagg	0.0001	0.0004	0.0004	0.0003
boost	0.0001	0.0002	0.0004	0.0001
em	**0.0000**	**0.0000**	**0.0000**	**0.0000**
epilink	0.0001	**0.0000**	0.0004	**0.0000**
febrl	0.0001	0.0003	0.0003	0.0001
libsvm	0.0003	0.0000	0.0004	0.0000

© Schattauer 2009

Downloaded by: Karolinska Institutet. Copyrighted material.

acc.	*(binary, dg)*	*(fuzzy, dg)*	*(binary, dz)*	*(fuzzy, dz)*
rpart	0.9994	0.9985	0.9996	0.9986
prune	0.9962	0.9985	0.9981	0.9986
bagg	0.9999	0.9996	0.9996	0.9997
boost	0.9999	0.9996	0.9996	0.9999
em	0.9998	1.0000	0.9998	**1.0000**
epilink	0.9999	1.0000	0.9996	0.9999
febrl	0.9998	0.9994	0.9996	0.9996
libsvm	0.9997	0.9998	0.9996	0.9998

Table 10 Accuracies for (K5, R20)

alpha	*(binary, dg)*	*(fuzzy, dg)*	*(binary, dz)*	*(fuzzy, dz)*
rpart	0.0024	0.0078	0.0002	0.0080
prune	0.0024	0.0078	0.0346	0.0080
bagg	**0.0002**	0.0005	**0.0002**	0.0007
boost	**0.0002**	0.0040	**0.0002**	0.0005
em	0.0058	0.0171	0.0452	0.0362
epilink	**0.0002**	0.0005	**0.0002**	0.0011
febrl	0.0024	0.0064	0.0024	0.0064
libsvm	**0.0002**	0.0027	**0.0002**	0.0027

Table 11 Alpha errors for (K5, SIM)

beta	*(binary, dg)*	*(fuzzy, dg)*	*(binary, dz)*	*(fuzzy, dz)*
rpart	0.0788	0.2508	0.0582	0.2319
prune	0.3125	0.2508	0.0581	0.2319
bagg	0.0255	0.2793	0.0584	0.2241
boost	0.0114	0.2214	0.0584	0.1440
em	0.0003	0.0003	**0.0002**	0.0004
epilink	0.0212	0.0223	0.0580	0.0265
febrl	0.0274	0.2356	0.0583	0.1012
libsvm	0.0328	0.0645	0.0585	0.2078

Table 12 Beta errors for (K5, SIM)

acc.	*(binary, dg)*	*(fuzzy, dg)*	*(binary, dz)*	*(fuzzy, dz)*
rpart	0.9259	0.7643	0.9454	0.7820
prune	0.7068	0.7643	0.9434	0.7820
bagg	0.9761	0.7380	0.9452	0.7898
boost	0.9893	0.7921	0.9452	0.8649
em	**0.9993**	0.9987	0.9970	0.9974
epilink	0.9801	0.9790	0.9456	0.9751
febrl	0.9741	0.7786	0.9451	0.9047
libsvm	0.9693	0.9393	0.9451	0.8050

Table 13 Accuracies for (K5, SIM)

the existence of some discrete stochastic model enables much more independence from specific characteristics of available training data. The stochastic record linkage based on the specific EM algorithm used here produced best classification results on real data and is only slightly inferior to *bagging*, *boosting* and *SVM* on artificial data. It should therefore be preferred for real world applications with similar data structures as considered in this paper.

Relying only on the structure of training data without assumption of an underlying random mechanism yields the best classification results when calibration and validation data have the same properties in every relevant aspect. In this case, the most elaborated methods considered (*bagging*, *boosting*, *SVM*) produce the best results. Whether the date of birth is entered en bloc or separately has no impact on best results. However, using string metrics enhances classification results in such cases.

Calibration of methods for object identification should be run in two phases. First, data pairs and corresponding comparison patterns are produced, and then a deterministic matching procedure is applied. In the first run, apparent duplicates and non-duplicates are identified. Decision trees can preliminary extract parts or all of the conditions for deterministic matching. This first phase is also an operationalization of the data universe from which data for object identification methods in the second phase are derived. When comparing different methods, it is necessary that their calibration data are equally filtered in order to ensure comparability. These data only consist of comparison pairs which are difficult to classify, and this reduces the amount of resources. After calibration, one has to determine when recalibration should be performed for iterative record linkage. Rules concerning recalibration should be based on the number of cases entered (e.g. after 10,000 cases have been entered). This two-phase process on the basis of EM has significantly improved the deterministic record linkage procedure used in the cancer registry from which our data originated.

We consider the following questions as essential for further research activities:

- What is the impact of leaving out or including additional attributes on classification results for the methods considered

Downloaded by: Karolinska Institutet. Copyrighted material.

© Schattauer 2009

in this paper? How can a procedure for feature selection in this context be set up?
- How can the optimal size of training data be determined?
- Can *bagging*, *boosting* and *SVM* be further enhanced in order to produce better results when training data and validation data don't have the same characteristics?
- Does another algorithm based on the EM principle with the assumption of a parametric probability function exhibit similar or even better results than the one implemented in this paper, or is the assumption of a discrete parametric form without a functional form crucial for good classification results?
- Which methods can produce better results than those considered in this paper?
- Does it suffice to use artificial data for calibrating methods in other contexts?

This outlook shows that the research area of record linkage as part of an applied science lacks the possibility of determining stable laws. This leads to the use of hand-tailored methods for special cases and the necessity to accept overfitting.

References

1. Fellegi IP, Sunter AB. A theory for record linkage. Journal of the American Statistical Association 1969; 64: 1183–1210.
2. Christensen R. Log-Linear Models and Logistic Regression. 2nd edition. New York: Springer; 1997.
3. Dempster AP, et al. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B 1977; 39: 1–38.
4. Winkler WE. Advanced methods for record linkage. Proceedings of the Section on Survey Research Methods; American Statistical Association; available at www.census.gov/srd/www/byyear.html; 1994.
5. Armstrong JA, Mayda JE. Model-based estimation of record linkage error rates. Survey Methodology 1994; 9: 137–147.
6. Haberman SJ. Log-linear fit for contingency tables-algorithm AS51. Applied Statistics 1972; 21: 218–225.
7. Espeland MA, Hui SL. A general approach to analyzing epidemiologic data that contains misclassification errors. Biometrics 1987; 43: 1001–1012.
8. Espeland MA, Odoroff C. Algorithms for computing maximum likelihood estimates from incomplete discrete data. Technical Report 01/84; University of Rochester (Statistical Department).
9. Darroch JN, Ratcliff D. Generalized iterative scaling for log-linear models. Annals of Mathematical Statistics 1972; 43 (5): 1470–1480.
10. Monge A, Elkan A. An efficient domain-independent algorithm for detecting approximately duplicate database records. In: Proc. of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. Tucson/Arizona; 1997.
11. Yancey WE. Evaluating string comparator performance for record linkage. Research Report available at www.census.gov/srd/www/byyear.html; 2005.
12. Jaro MA. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa/Florida. Journal of the American Statistical Association 1989; 89: 414–420.
13. Winkler WE. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. Proceedings of the Section on Survey Research Methods; American Statistical Association; 1990. pp 354–369.
14. Breiman L, Friedman, J Olshen R, Stone C. Classification and Regression Trees. Wadsworth; 1984.
15. Breiman L. Bagging predictors. Machine Learning 1996; 24: 123–140.
16. Freund Y, Schapire RE. Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference. San Francisco: Morgan Kaufman; 1996. pp 148–156.
17. Burges CJC. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 1998; 2 (2): 121–167.
18. Vapnik V. Statistical Learning Theory. John Wiley: New York; 1998.
19. Reng CM, Debold P, Specker C, Pommerening K. Generische Lösungen zum Datenschutz für die Forschungsnetze in der Medizin. München: MWV; 2006.
20. Christen P, Churches T, Hegland M. Febrl – a parallel open source data linkage system. In: Proceedings of the 8th Pacific-Asia Conference, Sydney 2004. New York: Springer Lecture Notes in Artificial Intelligence 2004; 3056: 638–647.
21. Quantin C, et al. The Epilink record linkage software. Methods Inf Med 2005; 44 (1): 66–71.
22. Haber E. AS207: Fitting a general log-linear model. Applied Statistics 1984; 33: 358–362.
23. Therneau TM, Atkinson EJ. An introduction to recursive partitioning using the rpart routine. Technical Report 61; Mayo Clinic, Section of Biostatistics; 1997.
24. Peters A, et al. ipred: Improved predictors. R News 2002. pp 33–36.
25. Chang CC, Lin CJ. Libsvm: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.
26. Lin CJ. On the convergence of the decomposition method for support vector machines. IEEE Transactions on Neural Networks 2001; 12 (6): 1288–1298.

Downloaded by: Karolinska Institutet. Copyrighted material.

© Schattauer 2009