A Novel Suite of Methods for Mixture Based Record Linkage[1]

Diego Zardetto[2], Monica Scannapieco[3]

Abstract

Record Linkage (RL) aims at identifying pairs of records coming from different sources and representing the same real world object. Despite several methods have been proposed to face RL problems, none of them seems to be at the same time fully automated and very effective. In this paper we present a novel suite of methods that instead possesses both these abilities. We adopt a mixture-model based approach, which structures a RL process into two consecutive tasks. First, mixture parameters are estimated by fitting the model to observed distance measures between pairs. Then, a probabilistic clustering of the pairs into Matches and Unmatches is obtained by exploiting the fitted model. In particular, we use a mixture model with component densities belonging to the Beta parametric family and we fit it by means of an original perturbation-like technique. Moreover, we solve the clustering problem according to both Maximum Likelihood and Minimum Cost objectives. To accomplish this task, optimal decision rules fulfilling one-to-one matching constraints are searched by a purposefully designed evolutionary algorithm. We present several experiments on real data that validate our methods and show their excellent effectiveness.

1. Introduction

Record Linkage (RL) (Elmagarmid et al., 2007) is the problem of identifying pairs of records coming from different sources and representing the same real world object. Integration of different data sources and improvement of the quality of single sources are only some of the real application scenarios that need to solve the RL problem. In Official Statistics, to cite just a single example, the need of performing a RL task arises whenever one tries to integrate statistical survey data with data coming from administrative archives, due to lacking or unreliable common record identifiers.

In this paper we present a novel suite of methods for RL, based on mixture models (McLachlan et al., 2000; McLachlan et al., 1988). These are statistical models that allow to represent a probability distribution as a convex combination of other distributions. As RL methods always rely on distance measures between record pairs, the intuition behind the use of mixtures models is that these observed distances arise from a superposition of two distinct probability distributions: the one stemming from the subpopulation of Matches (*M*) and the other from that of Unmatches (*U*). Evidently, the ultimate aim of such a statistical perspective on RL is to exploit the mixture model for classification purposes, i.e., to bring to light the hidden grouping of the pairs in the underlying *M* and *U* classes.

[1] This work is the outcome of a research collaboration and reflects opinions of both authors, however the primary contribution to the work has to be attributed to Diego Zardetto. This work was partially supported by Sapienza Università di Roma, "Progetti di Ricerca di Università" C26A074R53-C26A08L953.

[2] Cter (Istat), e-mail: zardetto@istat.it.

[3] Tecnologo (Istat), e-mail: scannapi@istat.it.

Since we formulate the RL problem as a classification problem driven by a mixture model, our approach requires the execution of two consecutive tasks. First, mixture parameters have to be estimated by fitting the model to the observed distance measures between pairs. Then, a probabilistic clustering of the pairs into Matches and Unmatches must be obtained by exploiting the fitted model.

The fitting step is the crucial one, as it implicitly determines the quality of the subsequent clustering results. However, it represents a very hard task; indeed, the problem of fitting a mixture model is always difficult, but it is even more severe in RL applications. This is due to the huge class-skew inherent in RL problems, where the very few (and unidentified) distance measures stemming from Matches risk to be completely overwhelmed by the bulk of those stemming from Unmatches. To overcome this difficulty we developed an original fitting technique inspired by Perturbation Theory (Bender et al., 1999). This technique allows us to obtain reliable estimates for the mixture parameters without relying on domain knowledge, thus not jeopardizing automation.

In the clustering step we use the fitted mixture model to search an optimal classification rule such that each pair can be assigned, based on its observed distance value, either to the *M* or to the *U* class. This is accomplished in such a way as to optimize a global objective function while satisfying a set of one-to-one matching constraints. These constraints arise when the data sets to be matched do not contain duplicates. In particular, we solve this constrained optimization problem by means of a purposefully designed evolutionary algorithm (Michalewicz, 1996). We use the algorithm to find decision rules that minimize either the probability of classification error (Maximum Likelihood objective) or, alternatively, the expected classification cost (Minimum Cost objective). The resulting rules critically depend on posterior estimates of class membership probabilities, obviously derived from the fitted mixture model.

The RL problem has received considerable attention by the scientific community, see (Elmagarmid et al., 2007) for a survey. Some works, e.g. (Chaudhuri et al., 2005; Guha et al., 2004), focus on solving the problem within a relational DBMS. Our approach is different from the cited ones because it is focused on effectiveness rather than on the ability to manage huge amount of data. The focus on effectiveness allows us to obtain results that are indeed superior to those obtained by (Chaudhuri et al., 2005) (whereas it was possible to compare the obtained results). However, our approach can be usefully inserted as a "decision engine" into a general RL system, even oriented to large databases.[4] Indeed this would just require to undertake a preliminary step dedicated to the reduction of the comparison space, for which several techniques have been proposed (Elmagarmid et al., 2007). This is because we designed our methods to be as general as possible; for instance, we do not rely on any restrictive assumption on the function to be used when comparing records.

Moreover, we remark the completely automated nature of our approach. This makes our work different on the one hand from supervised techniques for RL, e.g. (Tejada et al., 2001). On the other hand, our proposal is also different from the few unsupervised techniques, including (Chaudhuri et al., 2005; Verykios et al., 2000; Christen, 2007), none

[4] The practical feasibility of our methods when dealing with very large amounts of data will be tested in the Experiment section of the paper. There we shall face a big RL problem involving data collected in the Post Enumeration Survey carried out by the Italian National Institute of Statistics to estimate the coverage rate of the 2001 population Census.

of which, to the best of our knowledge, is fully automated. Indeed, though not requiring exactly a clerically prepared training set, such techniques still depend critically on some external inputs: e.g., human intervention is needed to set crucial parameters for the algorithms in (Chaudhuri et al., 2005) and (Christen, 2007), or to provide few labeled data in (Verykios et al., 2000).

Several works on RL rely on a probabilistic approach. A comparison with such works, mostly based on the Fellegi-Sunter formulation of the problem (Fellegi et al., 1969), will be presented later on in the paper (see Section 4), when we shall discuss the details of our proposal.

The paper is organized as follows. In Section 2 basic assumptions underlying statistical approaches to RL are introduced. These assumptions are then distilled in the form of a loose prior knowledge that our method is able to exploit successfully when facing practical RL tasks. Section 3 defines the adopted mixture model, whose component densities belong to the Beta parametric family. Section 4 is devoted to a thorough motivation and description of our original mixture-model fitting technique. Section 5 faces the clustering step, deals with one-to-one matching constraints and describes our clustering evolutionary algorithm. In Section 6 we test the effectiveness of our suite of methods on real-world RL instances. Finally, Section 7 draws some conclusions.

2. A Statistical Perspective on Record Linkage

Let us consider two sets $\mathcal{A}$, $\mathcal{B}$ of real world objects selected from a universe Ω and let us suppose that $\mathcal{A}$ and $\mathcal{B}$ contain some common objects, i.e. $\mathcal{A} \cap \mathcal{B} \neq \varnothing$. We denote by $\mathcal{M}$ the set of matched objects that appear in both $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{M} = \mathcal{A} \cap \mathcal{B}$, and by $\mathcal{U}$ the set of non-matched objects that appear in either $\mathcal{A}$ or $\mathcal{B}$ but not in both, $\mathcal{U} = (\mathcal{A} \cup \mathcal{B}) \setminus \mathcal{M}$. Obviously $\mathcal{A} \cup \mathcal{B} = \mathcal{M} \cup \mathcal{U}$.

We formally represent a *deterministic* data generating process as a mapping g from Ω to a data space S, such that $\Omega \ni \omega \overset{g}{\mapsto} g(\omega) = o \in S$. The image of $\mathcal{A}$ ($\mathcal{B}$) under g is a subset of S: we call it data set A (B). Notice that representing g as a mapping implicitly rules out the possibility that data sets A and B contain duplicated records. For *duplicates* we mean records that *i)* belong to the same data set and *ii)* correspond to the same real world object. We shall denote a real world object with a greek letter, $\alpha \in \mathcal{A}$, and the record to which it is mapped by g with the corresponding latin letter, $g(\alpha) = a \in A$.

The Record Linkage problem is defined as follows. Given two data sets A and B, find a partition of their cartesian product such that:

$$A \times B = M \cup U \tag{1}$$

where we introduced the set of record pairs that are Matches:

$$M = \{(a,b) \in A \times B : \alpha = \beta, \alpha \in \mathcal{A}, \beta \in \mathcal{B}\} \tag{2}$$

and the one of record pairs that are Unmatches:

$$U = \{(a,b) \in A \times B : \alpha \neq \beta, \alpha \in \mathcal{A}, \beta \in \mathcal{B}\} \tag{3}$$

Consider now an *ideal* (i.e. deterministic and error free) data generating process g_0, that is any *injective* mapping of Ω to *S*. If data sets *A* and *B* were generated under g_0, the RL problem would be trivial. Indeed, since $g_0(\alpha) = g_0(\beta) \Rightarrow \alpha = \beta$, the Matches set would be simply:

$$M = \{(a,b) \in A \times B : a = b\} \tag{4}$$

In other words, under an ideal data generating process the information stored into a record is *sufficient* to unambiguously identify the corresponding real world object.

Unfortunately, real world data generating processes are always affected by a wide variety of errors. Being the underlying error mechanism unknown (and hence the generated errors unpredictable), every real data generating process *g* has to be thought as a *stochastic* process. It can be useful to describe such a *g* as the addition of a random noise (representing the errors) to a signal (representing the records that would have been generated under ideal conditions): $g(\alpha) = \mathbf{n}(g_0(\alpha))$. Here the noise **n** is treated as a function of a deterministic argument (i.e. an input record $o \in S$) whose value is a "random variable" (i.e. an output random record $\tilde{o} = \mathbf{n}(o)$ still belonging to the data space *S*). We do not exclude that the real world data sets *A* and *B* have been generated by two distinct data generating processes, rather we only assume that the difference (if any) is entirely due to the error generating mechanism: $A = \mathbf{n}_A(g_0(\mathcal{A}))$, $B = \mathbf{n}_B(g_0(\mathcal{B}))$.

From a RL point of view, the main effect of the stochastic nature of a data generating process is that, with some nonzero unknown probabilities: *i)* the *same* real world object, $\mu \in \mathcal{M}$, can correspond to two *distinct* records, $m_A \neq m_B$ with $m_A \in A$ and $m_B \in B$; *ii)* two *distinct* real world objects, $\upsilon_1 \in \mathcal{U}, \upsilon_2 \in \mathcal{U}$, can correspond to the *same* record, $u \in A, u \in B$. Due to *i)* and *ii)* the set identity (4) is in general not true anymore and the RL problem becomes non-trivial.

Since simply assessing whether two records are *equal* is no longer sufficient to unambiguously classify the pair as a Match or as an Unmatch, let us introduce a "distance" function $d : S \times S \to \Re^+$. We do not require *d* to fulfill the triangle inequality, thus *S* (endowed with *d*) does not need to be a metric space. Instead, we do believe that *d* is able to capture reasonably well the "amount of difference" between two records (whatever their structure). Consider now a record pair (*a*, *b*) belonging to $A \times B$: since *a* and *b* are regarded as (realizations of) random variables, this will also hold true for their distance *d*(*a*, *b*). For the sake of simplicity (but without loss of generality as the distance is a bounded variable) we shall suppose *d* to be normalized so that its values fall inside the [0,1] interval.

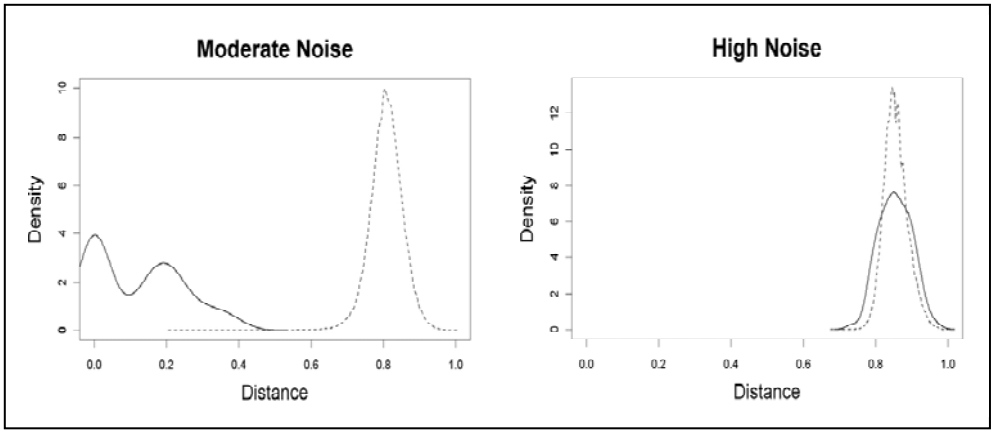
The basic idea behind every statistical approach to the RL problem is simple: we do believe that the observed distances between record pairs (although a priori unpredictable) carry some useful information about whether a given pair belongs to the set of Matches or to the one of Unmatches. Hence the distance *d* is viewed as an observable *auxiliary* random variable that we can use to infer the unknown outcomes of an (artificial) unobservable *interest* random variable *z*, namely the class membership indicator of a pair:

$$z_i = \begin{cases} 1 & \textit{if } i\textit{-th pair} \in M \\ 0 & \textit{if } i\textit{-th pair} \in U \end{cases} \tag{5}$$

Obviously this picture is founded on the hypothesis that the probability distribution of the distance *d* is significantly different inside the *M* and *U* classes. This in turn translates into the assumption that the noise component of the data generating process is only a small perturbation to the ideal signal.

Consider as an example Figure 1 where superimposed distance density plots are showed for Match (solid lines) and Unmatch (dashed lines) pairs. Both panels refer to the same original clean data sets to which we added random errors at moderate (left panel) and very high (right panel) rates.

Figure 1 - Distance density plots at moderate (left) and very high (right) error rates. Each graph shows superimposed density plots for Match (solid blue line) and Unmatch (dashed red line) pairs. [Physics data sets, see Section 6]

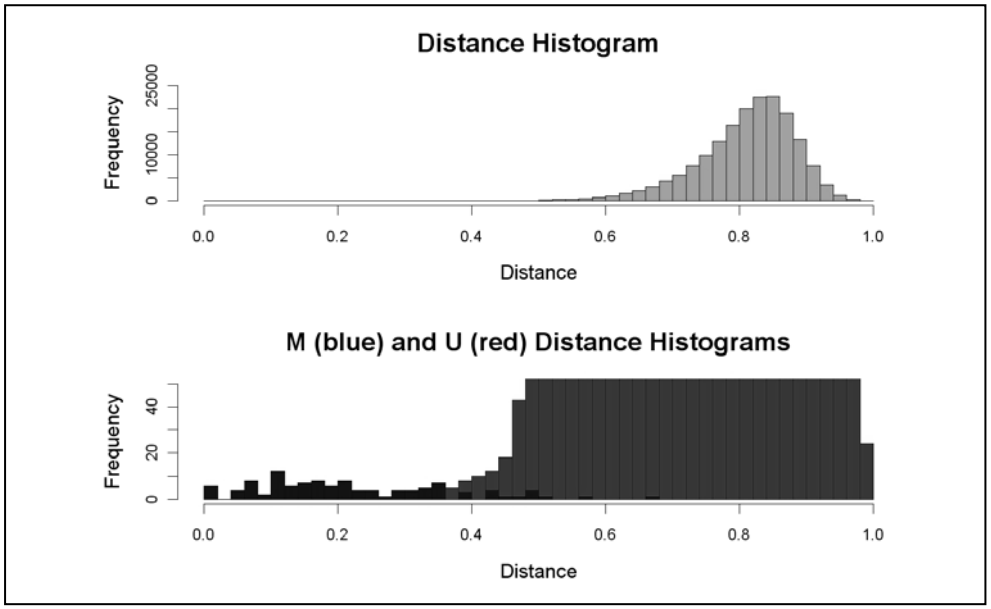


In the "moderate noise" scenario the shapes of the *M* and *U* distance densities are very different: Unmatches tend to be concentrated at higher distances than Matches, which furthermore still exhibit their own distinctive modal peak at zero distance; moreover *M* and *U* densities show only a relatively small overlap. On the contrary in the "high noise" scenario not only both Matches and Unmatches are located in the high-distance region, but in addition their densities almost completely overlap. While we are confident that a statistical approach to the RL problem will reveal itself appropriate to the first scenario, its use seems hopeless in the second. Luckily experience teaches two important lessons: *i)* the "high noise" scenario is almost never met in real-life applications, *ii)* the qualitative features of the *M* and *U* densities we just illustrated for the "moderate noise" scenario are quite general i.e. common to the great majority of practical problems. As a consequence we feel allowed to consider these main features as a *prior knowledge* about the underlying, unknown *M* and *U* distance probability distributions.

Besides this, it seems that another piece of *prior knowledge* is readily available, namely that Matches are rare. Indeed, if data sets *A* and *B* do not contain duplicated records (as we already assumed) the Match rate, i.e. the ratio between the cardinalities of *M* and $A \times B$, cannot exceed the value $1/\max(|A|,|B|)$. This value is very small in almost all the RL problems, even when blocking techniques have been applied.

Therefore, two distinct kinds of prior knowledge – named in the following $\mathcal{PK}1$ and $\mathcal{PK}2$ – are at our disposal: the first one, $\mathcal{PK}1$, concerning the main qualitative features of the *M* and *U* distance probability distributions, the second, $\mathcal{PK}2$, concerning the large class-skew between *M* and *U* pairs, with Matches being rare as compared to Unmatches. Figure 2 exemplifies in a clear-cut way the excellent agreement between the aforementioned basic assumpions $\mathcal{PK}1$ and $\mathcal{PK}2$ and sample data coming from a real-world RL instance, namely "Restaurants" (see Section 6).

Figure 2 - Pairwise-distances coming from a real-world RL instance [Restaurants data sets, see Section 6]. Upper panel: distance histogram of the whole unlabeled data (176,423 pairs). Lower panel: superimposed distance histograms for Match pairs (blue, dark) and Unmatch pairs (red, light); note that a 500 times y-axis zoom was needed to detect the feeble signal arising from the few Matches (112 pairs)



The Restaurants data contain only 112 Matches out of 331×533 = 176,423 pairs, yielding a Match rate of 6.3×10^{-4}, in full compliance with $\mathcal{PK}2$. Therefore, the histogram plotted in the upper panel, which represents the distance distribution of the whole unlabeled data (i.e. pairs belonging to both *M* and *U* classes), turns out to be totally dominated by the overwhelming contribution arising from Unmatches. As a consequence, a 500 times zoom of the low frequency region of the plot was needed in order to detect the feeble *M* distribution, as reported in the lower panel of figure 2. Moreover, the specific features of the *M* and *U* distance probabilities distilled into $\mathcal{PK}2$ are clearly reflected into the observed histograms. Indeed, Matches are dominating at low distances (with a bump at d = 0 arising from *M* pairs that haven't been hit by errors) and exhibit a soft right tail. Unmatches, in turn, dominate the high-distance region and show a soft left tail. Furthermore, there is only a small overlap between *M* and *U* tails.

We shall see in Sections 3 and 4 how our approach successfully exploits both $\mathcal{PK}1$ and $\mathcal{PK}2$ when facing practical RL tasks.

3. A Mixture Model for Record Linkage

Let us denote by n_A and n_B the cardinalities of data sets A and B respectively and by n_M and n_U the (unknown) cardinalities of classes M and U. Furthermore, let us call n_P the cardinality of the set of pairs $A \times B$ and define, for later convenience, $n_{min} = \min(n_A, n_B)$ and $n_{max} = \max(n_A, n_B)$, so that $n_P = n_A \cdot n_B = n_{min} \cdot n_{max}$. We shall denote the observed distances between record pairs by d_i where $i = 1, \ldots, n_P$ and an arbitrary ordering of the pairs is assumed. As we sketched in Section 2, we shall treat values d_i as n_P independent and identically distributed (iid) realizations of a random variable[5] d with values in [0,1].

We represent the probability density function (pdf) of d by the following two-component mixture density (McLachlan et al., 2000; McLachlan et al., 1988):

$$f(d) = \pi_M f_M(d) + \pi_U f_U(d) \tag{6}$$

where the components $f_{M,U}$ are the distance densities for the classes M and U and the mixing weights $\pi_{M,U}$ give the proportions of the classes, $\pi_{M,U} = n_{M,U}/n_P$ (so that $0 \le \pi_{M,U} \le 1$ and $\pi_M + \pi_U = 1$).

In what follows we assume that a suitable description of the mixture (6) can be achieved by supposing that its component densities belong to the *Beta* parametric family, namely $f_{M,U}(d) = \mathcal{Beta}(d; \theta_{M,U})$ where $\theta_{M,U} = (\alpha_{M,U}, \beta_{M,U})$ and:

$$\mathcal{Beta}(d; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} d^{\alpha-1}(1-d)^{\beta-1} \tag{7}$$

with Γ denoting the Euler Gamma function and the shape parameters fulfilling $\alpha > 0$ and $\beta > 0$. Hence density (6) is turned into a *parametric* two-component mixture model $f(d) = f(d; \Psi)$ described by five independent parameters $\Psi = (\alpha_M, \beta_M, \alpha_U, \beta_U, \pi_M)$.

The basic reasons that led us to select the Beta family for our mixture model can be summarized as follows:

- The Beta distribution has *support* in (0,1). This is appropriate for our distance random variable d.[6]
- The Beta distribution is *flexible*. By opportunely tuning its α and β parameters the Beta density can take a broad range of shapes, such as: flat, U-shape (as well as inverse U-shape), J-shape (as well as mirrored J-shape), unimodal (both narrowly peaked or smooth), symmetrical or asymmetrical, and so on.
- The Beta distribution can be *skewed*, both positively and negatively. This property is desirable as we expect from the discussion about $\mathcal{PK1}$ in Section 2 that the distance densities for M and U classes are skewed (with longer right and left tail respectively).

[5] For economy of notation random variables will not be typographically distinguished from their realizations: the intended meaning should be clear from the context.

[6] There is a technical subtle point arising from the fact that the support of the Beta distribution does *not* include the boundary values $d = 0$ and $d = 1$, whereas pairwise distances equal to 0 or 1 can be (and in general are) observed in practice. Due to space limitations, we cannot describe in a detailed way how we solved this problem; we just point out that, instead of a mixture of simple Betas, we adopted a mixture of *Inflated Beta* distributions (Ospina et al, 2010).

- The parameters controlling the Beta distribution are *shape parameters*. This is an important point because it allows to easily translate our prior knowledge $\mathcal{PK}1$ into a well defined set of constraints acting on the parameter space (this would not have been the case if, instead, we decided to plug into our mixture model (6) some pdfs described by location and scale parameters).

We stress here that the primary aim of our mixture model is *not* the one of obtaining a *high-quality* description for the observed distance distribution of the d_is. If it were the case, one could be led to model a mixture with more than two components or, otherwise, to worry about the inability of the Beta family to represent multimodal distributions (look e.g. at the shape of the *M* density in the left panel of Figure 1). On the contrary we basically see the mixture as a device to *exploit* the observed d_is distribution in order to bring to light the hidden underlying grouping of the record pairs into the *M* and *U* classes. Since, at least in a mixture approach mainly aimed at *clustering*, it is the fit of the tails of the class distributions that turns out to be crucial (McLachlan et al., 1988) (rather than the fit of the main body of the data, for which a good "average description" seems enough), our choice of the Beta family will prove to be satisfactory.

Once equipped with our two-component Beta mixture model, we have to accomplish two major tasks in order to find a solution to a specific RL problem:

1. We must fit the mixture model to the observed distance values d_is, so as to obtain estimates $\hat{\Psi}$ for the mixture parameters.
2. We must achieve (upon plugging into the model the estimates $\hat{\Psi}$) a probabilistic clustering of the record pairs tied to the observed distances d_is into classes *M* and *U*.

The task of fitting a mixture model can be handled by a variety of methods, including e.g. the method of moments (MOM), Bayesian methods and graphical methods. In the present work we chose the Maximum Likelihood (ML) method which is by far the most popular. Anyway, the technique we developed for obtaining ML estimates for the parameters of our mixture model is indeed original. Therefore, the next section of the paper will be devoted to a detailed motivation and description of this technique.

Section 5 will describe the clustering task, which we shall face in a decision-theoretic framework. An *optimal* classification rule will be searched such that each record pair *i* can be assigned, based on its observed distance value d_i, either to the *M* or to the *U* class, in such a way as to optimize a given *global* objective function.

Before going into further details, we observe – as a general remark – that our methods can be ascribed as a whole to the frequentist inferential scheme. This seems to be the case also for most of the classical papers in the probabilistic RL literature. On the other side, some authors have proposed techniques to face the RL problem in a purely Bayesian framework, see e.g. (Fortini et al., 2001; Larsen, 2005). Although a discussion of such techniques is beyond the scope of the present paper, future works could explore the possibility to embed our strategies (e.g. the way we use $\mathcal{PK}1$ and $\mathcal{PK}2$) in a purely Bayesian inferential scheme.

4. Fitting the Mixture Model: a Perturbation-like Approach

Under our two-component Beta mixture model, the log-Likelihood associated to the observed distance values $\mathbf{d} = (d_1, \ldots, d_{n_P})$ reads:

$$\log \mathcal{L}(\mathbf{d}; \Psi) = \sum_{i=1}^{n_P} \log\left[\pi_M f_M(d_i; \theta_M) + \pi_U f_U(d_i; \theta_U)\right] \tag{8}$$

The maximum Likelihood estimator (MLE) of the unknown mixture parameters Ψ can thus be defined as follows:

$$\hat{\Psi} = \text{argmax}_{\Psi}\left[\, \log \mathcal{L}(\mathbf{d}; \Psi) \,\right] \tag{9}$$

Finding the MLE of a model is, with few exceptions, a hard task. Not only an analytic closed-form solution to the problem is generally not available, what's more maximizing the Likelihood by means of numerical routines turns out to be difficult for most of the general-purpose optimization algorithms. It is not by chance that an ad-hoc class of optimizers, namely the Expectation-Maximization (EM) (Dempster et al., 1977) family, has been specifically tailored to handle ML problems. Unfortunately, the situation is even worse for *mixture* models, since their Likelihood function is often unbounded over the parameter space and typically exhibit many spurious local maxima. Anyway, the EM (or an EM-like) algorithm remains by far the favorite tool to handle ML estimation of mixture model parameters.

Many authors (Jaro, 1989; Armstrong et al., 1992; Winkler, 1993; Winkler, 1994; Belin et al., 1995; Larsen et al., 1997; Larsen et al., 2001) have already proposed the use of mixture models to solve RL problems. Some comments are in order, since the method we developed to fit our mixture is significantly different from the ones implemented in those classical papers:

- They all exploit mixture models adopting a *Fellegi-Sunter* approach (Fellegi et al., 1969). Thus the role of our auxiliary unidimensional distance variable *d* is typically played by a *k*-vector variable whose components represent agreement/disagreement outcomes obtained when comparing record pairs on *k* matching fields.[7]
- With the only exception of (Armstrong et al., 1992), they all use an EM-like algorithm in the mixture model fitting phase.
- A mixture model with more than two components is generally selected. From a clustering point of view, this obviously raises the question of how many and which mixture components have to be associated to each of the *M* and *U* classes.

[7] On the whole, i.e. when taking into account the methods we propose for *both* the *fitting phase* (Section 4) and the *clustering phase* (Section 5), our approach differs very much from Fellegi-Sunter's one (FS). Just to mention some of the differences: *i)* FS has a third class (besides *M* and *U*), namely the *Possible Match* class; *ii)* FS relies on a completely different notion of "optimal decision rule", which is neither based on Maximum-Likelihood nor on Minimum-Cost, but rather involves the *Possible Match* class; *iii)* FS implementations tipically rely on the assumption of conditional-independence for the components of the comparison vector; *iv)* FS applications generally ask the user to set classification thresholds.

- Their methods always encompass some amount of "human work" which is specifically meant to incorporate some kind of "experience" into the mixture fit. Possible ways to achieve this result are the following: *i)* guesses based on previous linkage applications on similar data can be employed to build initial estimates of model parameters; *ii)* selected record pairs (typically the more difficult to classify) can be sent to clerical review and the mixture model can be re-fitted by using also those clerically classified cases; *iii)* a training set of pre-labeled record pairs can be prepared, in order to fit the mixture component(s) describing e.g. the *M* class by means of only cases surely belonging to that class.

With regard to last point, the cited papers agree on the following finding: as far as RL problems are concerned, mixture models tend to cluster the pairs into groups that, despite achieving a good fit, often do *not* correspond to the desired *M* and *U* classes. The reason for this behavior seems more controversial, with some authors ascribing it mainly to model misspecification and some others putting the emphasis on the difficulty in the estimation of mixture model parameters. Anyway, a sharp picture emerges from the literature above: the only way a mixture model can yield high-quality RL results is to incorporate in it some kind of "previous knowledge". We agree on this last conclusion, but we contend that the necessary prior knowledge can be incorporated *without* relying on "clerical work", thus not jeopardizing automation.

In our opinion the poor *clustering* results that a "basic" (i.e. not experience-enriched) mixture-model would generally achieve in RL applications have to be mainly imputed to the huge class-skew inherent in these problems. More specifically, we argue that troubles would usually arise from the previous model *fitting* phase, due to the extreme Match rarity. Indeed, unless some countermeasure is adopted, whatever fitting algorithm would tend to tune *all* the model parameters so as to better describe some peculiar feature of the dominating *U* distance distribution.

The fitting technique we propose tries to exploit our two-fold previous knowledge ($\mathcal{PK}1$ and $\mathcal{PK}2$) in order to prevent the few (and so far unidentified) distance values stemming from the *M* class from being completely overwhelmed by those belonging to the *U* class. This is accomplished by means of a *Two-Step* algorithm. Before going into details, we offer here an intuitive insight into its working mechanism. The *First-Step* concentrates on the *U* component mixture parameters and is specifically aimed at "factorizing" the leading contribution arising from Unmatches. The *Second-Step* strives to increase the Likelihood achieved in the previous step by using the remaining mixture parameters in a "smart way"; that is, *M* density parameters are tuned in such a way as to better fit the behavior of the distance distribution *exactly* in those regions of the [0,1] interval in which values stemming from Matches are more likely to be found.

Our two-step algorithm follows a *perturbation-like* approach to the ML mixture fitting problem (9). *Perturbation Theory* (Bender et al., 1999) is a family of mathematical methods aimed at finding an approximate solution to problems that cannot, in general, be solved exactly *but* would become easy to solve if a parameter, say ε, had a given value, say $\varepsilon = 0$. The key idea is to build an approximation to the unknown solution of the true (i.e. $\varepsilon \neq 0$) problem by *perturbing* the known solution of the easier (i.e. $\varepsilon = 0$) problem, that is by adding to it further terms. These "higher orders" terms can be computed iteratively by some systematic procedure and, if $\varepsilon < 1$, turn out to be suppressed by increasing powers of ε. As a consequence, a

satisfactory solution can very often be obtained by truncating the series at its second term, that is by retaining only the initial solution and the first-order perturbative correction.

Due to prior knowledge $\mathcal{PK}2$, we are aware that the true unknown value of the mixing weight π_M is very small:

$$\pi_M \leq \frac{1}{n_{max}} << 1 \tag{10}$$

Moreover, in the limit $\pi_M = 0$, our hard *mixture* MLE problem (9) is turned into the much simpler problem of finding the MLE for the *U* density *alone*. Therefore we are allowed to look for a perturbative expansion of our mixture model parameters in powers of π_M:

$$\theta_U = \theta_U^{(0)} + \theta_U^{(1)}\pi_M + \theta_U^{(2)}\pi_M^2 + \cdots \tag{11}$$

$$\theta_M = \theta_M^{(0)} + \theta_M^{(1)}\pi_M + \theta_M^{(2)}\pi_M^2 + \cdots \tag{12}$$

where the $^{(j)}$ superscript denotes the *j*-th-order coefficient to be estimated. By inserting expansions (11) and (12) inside (9) we get a hierarchy of sub-problems that can be solved in a chain to yield the desired estimates $\hat{\theta}_{M,U}^{(j)}$. As we are going to see, our First-Step and Second-Step optimizations are respectively in charge of solving the zeroth-order and first-order approximations of problem (9). Moreover, Second-Step optimization has also to incorporate the first piece of prior knowledge we collected, i.e. $\mathcal{PK}1$.

4.1 First-Step Optimization

To zeroth-order in perturbation theory our MLE problem reads:

$$\log \mathcal{L}_I(\mathbf{d}; \theta_U^{(0)}) = \sum_{i=1}^{n_P} \log\left[f_U(d_i; \theta_U^{(0)}) \right] \tag{13}$$

$$\hat{\theta}_U^{(0)} = \underset{\theta_U^{(0)}}{\operatorname{argmax}} \left[\log \mathcal{L}_I(\mathbf{d}; \theta_U^{(0)}) \right] \tag{14}$$

where our First-Step *effective Likelihood* $\mathcal{L}_I$ differs from the real Likelihood $\mathcal{L}$ by terms that are at most of order π_M.

Since the mixture structure has disappeared from (13), an EM-like optimizer is no longer a mandatory choice. Indeed, when implementing the method, we chose to maximize $\mathcal{L}_I$ by means of a *faster* quasi-Newton[8] multivariate optimization algorithm. The starting guess needed to initialize the optimizer was computed as the MOM estimator of parameters $\theta_U^{(0)} = (\alpha_U^{(0)}, \beta_U^{(0)})$ given the observed distance distribution, namely:

[8] This required to derive the analytical expression of the gradient of the effective log-Likelihood (13), which we cannot report here due to space limitations.

$$\alpha_U^{(0)}\Big|_{start} = \left[\frac{\bar{d}(1-\bar{d})}{v} - 1\right]\bar{d} \tag{15}$$

$$\beta_U^{(0)}\Big|_{start} = \left[\frac{\bar{d}(1-\bar{d})}{v} - 1\right](1-\bar{d}) \tag{16}$$

where the sample mean $\bar{d} = (1/n_P)\sum_{i=1}^{n_P} d_i$ and the sample variance $v = (1/n_P)\sum_{i=1}^{n_P}(d_i - \bar{d})^2$ of the d_is have been used.

Coming back to the method, once a solution for the First-Step problem (14) is at hand, it becomes possible to take a step further in the perturbative approximation of the original MLE problem. This is accomplished by the subsequent Second-Step optimization.

4.2 Second-Step Optimization

If, after having inserted expansions (11) and (12) inside (9), we keep terms up to first-order in π_M and, in addition, we use the achieved zeroth-order solution (14), we get:

$$\log \mathcal{L}_{II}(\mathbf{d}; \theta_M^{(0)}, \pi_M) = \sum_{i=1}^{n_P} \log\left[\pi_M f_M(d_i; \theta_M^{(0)}) + (1-\pi_M) f_U(d_i; \hat{\theta}_U^{(0)})\right] \tag{17}$$

where our Second-Step *effective Likelihood* $\mathcal{L}_{II}$ differs from the real Likelihood $\mathcal{L}$ by terms that are at most of order π_M^2.

It is worth noting that, despite contributions of order π_M have been retained, equation (17) does *not* contain the first-order coefficient $\theta_U^{(1)}$ of (11), that is $\mathcal{L}_{II}$ does *not* depend on U parameters. This is a direct consequence of $\hat{\theta}_U^{(0)}$ being a (local) maximum of $\log \mathcal{L}_I$. Indeed, as the gradient of $\log \mathcal{L}_I$ is zero in $\hat{\theta}_U^{(0)}$, a deviation of order π_M from $\hat{\theta}_U^{(0)}$ can have at most an effect of order π_M^2 on the log-Likelihood. Therefore, everything goes as if we were now switching on the parameters describing the M component of the mixture, while those from the U component have been "frozen" to the estimated values found in the previous optimization step.

We must now incorporate our prior knowledge $\mathcal{PK}1$ and $\mathcal{PK}2$ inside the Second-Step optimization problem. Since equation (10) already summarizes $\mathcal{PK}2$, we have only to translate $\mathcal{PK}1$ into a set of constraints acting on M parameters $\theta_M^{(0)} = (\alpha_M^{(0)}, \beta_M^{(0)})$. This task can be completed by studying the dependence of the Beta distribution (7) on the shape parameters and by exploiting known formulae for its first moments; the following chain of translations results:

1. Unmatches are mainly located in the high distance region and the U density is negatively skewed:
$$\beta_U < \alpha_U$$
2. Matches are mainly located in the low distance region and the M density is positively skewed:
$$\beta_M > \alpha_M$$

3. The U density dominates the M density in the limit $d \to 1$:

$$\beta_M > \beta_U$$

4. The M density dominates the U density in the limit $d \to 0$:

$$\alpha_M < \alpha_U$$

5. The M and U densities have a small overlap. The easiest way to express this somewhat fuzzy requirement, while fulfilling constraints 1) – 4), is as follows:

$$\alpha_M < \beta_U \quad and \quad \beta_M > \alpha_U$$

Replacing inside 1) – 5) U parameters with estimates and neglecting redundant constraints, we eventually obtain the complete formulation of the Second-Step MLE problem:

$$\{\hat{\theta}_M^{(0)}, \hat{\pi}_M\} = \underset{\{\theta_M^{(0)}, \pi_M\}}{\text{argmax}} \left[\log \mathcal{L}_{II}(\mathbf{d}; \theta_M^{(0)}, \pi_M) \right] \tag{18}$$

$$subject\ to:$$

$$\alpha_M^{(0)} < \hat{\beta}_U^{(0)} \tag{19}$$

$$\beta_M^{(0)} > \hat{\alpha}_U^{(0)} \tag{20}$$

$$\pi_M \leq 1/n_{max} \tag{21}$$

It should be noted that, since $\pi_M = 0$ is a feasible point for the constrained ML problem (18)-(21), and as $\mathcal{L}_{II}(\mathbf{d}; \theta_M^{(0)}, \pi_M = 0) \equiv \mathcal{L}_I(\mathbf{d}; \hat{\theta}_U^{(0)}) \quad \forall \theta_M^{(0)}$, the following inequality will be satisfied: $\max(\log \mathcal{L}_{II}) \geq \max(\log \mathcal{L}_I)$. Consequently, Second-Step Optimization cannot decrease the Likelihood achieved in the First-Step, but rather will in general *increase* it. This is coherent with the quick outline we gave in Section 4 on our Two-Step perturbative fitting technique.

Furthermore, thanks to the decoupling of U and M parameters, once again the need of an EM-like optimizer has been overcome. Indeed, the software we developed solves the *constrained* ML problem (18)-(21) by means of a box-constrained quasi-Newton[9] multivariate algorithm. Besides the usual by-product of saving computation time, this choice freed us from the tricky machinery required to manage a *constrained* EM-like algorithm (Winkler, 1993). When testing our application, random starting values drawn from the feasible region (19)-(21) were used to initialize the optimizer. Because fairly stable results were found, we eventually fixed the starting guess as follows:

[9] Again, this required to derive the analytical expression of the gradient of the effective log-Likelihood (17), which we cannot report here due to space limitations.

$$\alpha_M^{(0)}\Big|_{start} = \hat{\beta}_U^{(0)}/10 \tag{22}$$

$$\beta_M^{(0)}\Big|_{start} = 10\hat{\alpha}_U^{(0)} \tag{23}$$

$$\pi_M\Big|_{start} = (1/2)(1/n_{max}) \tag{24}$$

Computing a solution of the Second-Step optimization problem (18)-(21) completes the task of fitting our two-component Beta mixture model. By plugging into the model the achieved estimates:

$$\hat{\Psi} = (\hat{\alpha}_M^{(0)},\ \hat{\beta}_M^{(0)},\ \hat{\alpha}_U^{(0)},\ \hat{\beta}_U^{(0)},\ \hat{\pi}_M) \tag{25}$$

we can now switch to the problem of *clustering* the record pairs into classes *M* and *U*.

5. Clustering Pairs using the Mixture Model

As we already mentioned in Section 3, our mixture model has been specifically designed to be used for clustering purposes. The goal is, indeed, to exploit the model – along with its ML estimated parameters (25) – to assign each record pair *i* either to the *M* or to the *U* class. Clearly the obtained classification for the *i*-th pair will depend on its observed distance value d_i.

5.1 Optimal Classification Rules from Decision Theory

Let us attach to each record pair *i* a class membership indicator z_i with value 1 if the pair is a Match and 0 otherwise. The *true* value of z_i is obviously *unknown*: it will precisely represent the target of our inferences. We shall, consequently, treat variables z_i as *iid*[10] realizations of a *latent* random variable *z*. Variable *z* can be incorporated inside our mixture model (6), which describes the distance pdf, by assuming that:

1. Variable *z* is distributed according to a single draw from a Binomial distribution with success probability given by π_M.
2. The conditional densities of *d*, given *z* = 1 and *z* = 0, are $f_M(d;\theta_M)$ and $f_U(d;\theta_U)$ respectively.

Under conditions 1) and 2), the *complete* mixture density can be expressed as follows:

$$g(d,z;\Psi) = \left[\pi_M f_M(d;\theta_M)\right]^z \left[\pi_U f_U(d;\theta_U)\right]^{1-z} \tag{26}$$

Correspondingly, the *complete* (and *unobservable*) log-Likelihood associated to the observed distance values $\mathbf{d} = (d_1,\ldots,d_{n_P})$ and to the hidden class labels $\mathbf{z} = (z_1,\ldots,z_{n_P})$ reads:

[10] Notice that the independence assumption on variables z_i cannot hold true for 1:1 RL problems (i.e. when the data sets to be matched do not contain duplicates). We shall come back to this issue in Section 5.2.

$$\log \mathcal{L}^c(\mathbf{d},\mathbf{z};\Psi) = \sum_{i=1}^{n_P} \log\left[g(d_i,z_i;\Psi)\right] =$$
$$= \sum_{i=1}^{n_P} \log\left[\pi_U f_U(d_i;\theta_U)\right] + \sum_{i=1}^{n_P} z_i \log\left[\frac{\pi_M f_M(d_i;\theta_M)}{\pi_U f_U(d_i;\theta_U)}\right] \quad (27)$$

Thanks to (26), the mixing weights π_M and π_U can now be understood as the *prior probabilities* that the *i*-th pair belongs to class *M* and *U* respectively, while the corresponding *posterior probabilities*, given the distance value d_i observed for the pair, are:

$$\tau_i^C(d_i;\Psi) = \pi_C f_C(d_i;\theta_C)/f(d_i;\Psi) \qquad C = \{M,U\} \quad (28)$$

Estimates of posterior probabilities $\hat{\tau}_i^C$ can be built by simply plugging into (28) the previously computed estimates (25) for the model parameters $\hat{\Psi}$. As we are going to see, these values $\hat{\tau}_i^C$ play a central role in the clustering task.

A "classification rule" that assigns each record pair *i* to a class is nothing but a rule to *infer* a value $\hat{z}_i$ for the hidden variable z_i. An *optimal* rule has moreover to work in such a way as to optimize some global objective function.

A first, very natural choice is to select as objective function the complete log-Likelihood itself. This means that we look for an allocation vector $\hat{\mathbf{z}}$ that maximizes the complete data Likelihood under the model (26), namely:

$$\hat{\mathbf{z}} = \text{argmax}_{\mathbf{z}}\left[\log \mathcal{L}^c(\mathbf{d},\mathbf{z};\hat{\Psi})\right] \quad (29)$$

The solution of (29) follows easily from the structure of (27):

$$\hat{z}_i = \begin{cases} 1 & if \quad \hat{\tau}_i^M \geq \hat{\tau}_i^U \\ 0 & otherwise \end{cases} \quad (30)$$

Formula (30) is a classical Decision Theory result (Duda et al., 2000), known as the "Bayes decision rule" (or as "Maximum a Posteriori (MAP) rule"): it assigns each pair to the class to which the pair has the highest estimated posterior probability of belonging.

A sometimes useful alternative is to find the classification rule that minimizes the expected value of a *Loss Function*. For instance, different costs can be assigned to the possible outcomes of a decision (Verykios et al., 2003). Indeed, the cost C_{PU} for declaring a pair to be a Match (we call this a *positive* decision) when it is actually an Unmatch can differ from the cost C_{NM} for declaring a pair to be an Unmatch (we call this a *negative* decision) when it is actually a Match. In this situation, an appropriate Loss Function would be the expected *Total Cost* associated to a classification $\hat{\mathbf{z}}$:

$$C_{TOT}(\mathbf{d},\hat{\mathbf{z}};\hat{\Psi}) = \sum_{i=1}^{n_P}\left[C_{PM}\hat{\tau}_i^M + C_{PU}\hat{\tau}_i^U\right]\hat{z}_i + \sum_{i=1}^{n_P}\left[C_{NM}\hat{\tau}_i^M + C_{NU}\hat{\tau}_i^U\right](1-\hat{z}_i) \quad (31)$$

Minimizing the expected Total Cost (31) with respect to $\hat{\mathbf{z}}$ is straightforward and leads to the following optimal decision rule:

$$\hat{z}_i = \begin{cases} 1 & if \quad (C_{NM} - C_{PM})\,\hat{\tau}_i^M \geq (C_{PU} - C_{NU})\,\hat{\tau}_i^U \\ 0 & otherwise \end{cases} \tag{32}$$

Once again, formula (32) is a classical Decision Theory result (Duda et al., 2000). If a zero cost is assigned to a correct decision (i.e. $C_{PM} = C_{NU} = 0$) and the same cost, say one, is assumed for both kinds of wrong decisions (i.e. $C_{PU} = C_{NM} = 1$), the expected Total Cost (31) simply measures the expected number of classification errors; accordingly, the optimal decision rule (32) is turned into the Bayes rule (30).

The software we developed is able to cluster the pairs according to both (30) and (32) rules. Anyway, these clustering results have to be understood only as *"provisional solutions"* to the RL problem at hand. Indeed, while building our mixture fitting method in Section 4, we assumed – recall e.g. equation (21) – that the data sets to be matched did *not* contain duplicates. This obviously translates into a set of one-to-one matching constraints that, in general, are *not* fulfilled by the aforementioned optimal decision rules. Next section is devoted to describe how our method overcomes this difficulty.

5.2 Dealing with One-to-One Matching Constraints

As data sets *A* and *B* do not contain duplicates, each record belonging to either data set can match at most a single record selected from the other data set. Therefore, the number of true Matches cannot exceed the cardinality of the smaller data set, i.e. $n_M \leq n_{min}$, whence equations (10) and (21) follow.

In order to express these one-to-one (1:1) matching constraints in a formal way, let us switch to a more convenient matrix notation. We start by arranging the n_P observed distance values d_i into a $n_{min} \times n_{max}$ matrix D, in such a way that element D_{ij} represents the distance between the *i*-th record of the smaller data set and the *j*-th record of the bigger data set. Next we introduce matrices Z and $\hat{Z}$ to store, in the same way, the unknown class membership indicators of the pairs and their inferred values, respectively. Accordingly, quantities depending on features of the generic *i*-th pair (like d_i, z_i, $\hat{z}_i$, $\hat{\tau}_i^C$, and so on) have to be replaced, inside all previous sections formulae, by the corresponding two-index quantities (like D_{ij}, Z_{ij}, $\hat{Z}_{ij}$, $\hat{\tau}_{ij}^C$, and so on). At the end, 1:1 matching constraints can be readily incorporated into the problem of finding an optimal decision rule. For instance, the ML problem (29) now reads:

$$\hat{Z} = \operatorname{argmax}_Z \left[\log \mathcal{L}^c(D, Z; \hat{\Psi}) \right] \tag{33}$$

subject to:

$$\sum_{j=1}^{n_{max}} Z_{ij} \leq 1 \quad \forall i \tag{34}$$

$$\sum_{i=1}^{n_{min}} Z_{ij} \leq 1 \quad \forall j \tag{35}$$

with (34) and (35) obviously implying $\sum_{ij} Z_{ij} \leq n_{min}$.

Problem (33)-(35) deserves some comments. First of all, it is apparent that, due to the 1:1 restrictions (34) and (35), variables Z_{ij} cannot be *stochastically independent* (e.g. if $Z_{km} = 1$ then $Z_{im} = 0 \ \forall i \neq k$ and $Z_{kj} = 0 \ \forall j \neq m$). As a consequence, being the underlying *iid* property violated, one should not use formula (27) to express the complete data Likelihood. Anyway, the task of deriving its correct expression without relying on the *iid* assumption turns out to be too difficult to be handled. Therefore, one is led to the practical compromise of solving the constrained problem (33)-(35) while keeping the old formula (27) for the complete data Likelihood.

In any case, 1:1 matching constraints heavily affect the complexity of both ML and Minimum Cost optimization problems: now, indeed, a decision taken on a pair *influences* decisions to be taken on other pairs. Moreover, clustering results based on classical decision rules (30) and (32) will not, in general, fulfill 1:1 constraints. Consequently, our software treats these quickly computed results as *"provisional solutions"*. This means that they are checked against (34) and (35), and, only if 1:1 constraints happen to be already satisfied, they are retained as *"definitive results"*. When, on the contrary, 1:1 constraints turn out to be broken, *"definitive results"* are searched by facing directly the constrained optimization problem (33)-(35) (or its Minimum Cost counterpart). This new – and *harder* – clustering task is accomplished by means of a purposefully designed Evolutionary Algorithm (EA) (Michalewicz, 1996). Even in this case the work carried out to compute the *"provisional solutions"* will not get wasted, as useful pieces of information stemming from these solutions will be exploited by the clustering EA.

Before going into further details, we briefly argue why we chose an EA. A few papers from the RL literature (Jaro, 1989; Winkler, 1994) tackled the 1:1 problem[11] by using Simplex-based algorithms. Indeed, since both the objective function and the constraints are linear in Z_{ij}, equations (33)-(35) can be formulated as a Binary Linear Programming (BLP) problem. The main concern with this approach is tied to memory usage. As a matter of fact, if one denotes with n the size of the data sets to be matched (i.e. $n_A \approx n_B \approx n$), one sees that the number of unknowns and inequalities for the BLP problem grow like n^2 and n, respectively. The net result is that, due the heavy memory overhead of a Simplex-based solver, the BLP approach cannot be applied to real-world data sets *unless* a very efficient previous blocking step has been performed. On the contrary, as we shall see in Section 5.3, the size of the biggest data structure stored by our EA grows almost linearly with n. As a consequence, our software was able to handle all the RL problems listed in Section 6 (with the obvious exception of the huge $\mathbf{PES_{full}}$ instance, see the discussion therein) by running in an ordinary PC environment and without relying on blocking. Finally, we observe that our EA could be readily applied to clustering tasks that involve more complex Loss Functions than (32), e.g. nonlinear Loss Functions.

[11] Notice, however, that those authors simply force 1:1 clustering restrictions on a statistical model that does not encompass them. On the contrary, we already took into account 1:1 matching constraints while fitting our mixture model parameters (see equation (21)).

5.3 An Evolutionary Algorithm for 1:1 Clustering

The Evolutionary optimization metaheuristic is so versatile that EAs can often be employed to find a satisfactory solution even for problems for which no other solution strategy is known. This extreme flexibility has two major prices: i) no standard design rules for EAs are available; ii) EAs performance critically depend on how smartly problem-specific pieces of information are incorporated into the algorithm. Considerations i) and ii) should push us to provide a thorough justification for each aspect of our clustering EA. Nevertheless, due to space limitations, we shall restrict ourselves to a very concise outline.[12] In what follows we list the pseudocode of the algorithm and quickly describe basic choices, parameters and operators.

EA PSEUDOCODE

```
EA[n_ind, n_gen, p_muta, g_stall]
g ← 0
generate Initial Population[n_ind]
compute Fitness
while (¬ Termination Criterion[n_gen, g_stall]) do
  g ← g + 1
  apply Selection
  apply Reproduction + Repair
  apply Mutation[p_muta]
  compute Fitness
end while
return Best Fit individual found
```

Search Space. The EA search space is the set of all the $n_{min} \times n_{max}$ matrices Z with $\{0,1\}$ elements that fulfill 1:1 constraints (34) and (35). It is a *huge* search space whose cardinality is given by:

$$\mathcal{N}_Z = \sum_{k=0}^{n_{min}} \binom{n_{max}}{k} \binom{n_{min}}{k} k!$$

(just to get an impression: if $n_{max} = n_{min} = 150$ then $\mathcal{N}_Z = 1.28 \times 10^{272}$).

Representation. We encode a generic candidate solution Z (*phenotype*) by means of a vector ζ of length n_{min} (*genotype*). Elements of ζ (*alleles*) can be 0 or integers between 1 and n_{max}, namely $\zeta_k \in \{0,1,\ldots,n_{max}\}$ with $k = 1, 2, \ldots, n_{min}$. The meaning of the alleles is easily understood. If $\zeta_k = 0$, then the candidate solution states that the k-th record of the smaller data set *does not match any record* of the bigger data set. If, on the contrary, $\zeta_k = j > 0$, then the candidate solution states that the k-th record of the smaller data set

[12] We assume a basic knowledge of EAs and refer to (Michalewicz, 1996) (and references therein) for more advanced topics.

does match the *j*-th record of the bigger data set. Obviously a *legal* genotype, that is a genotype encoding a *feasible* candidate solution *Z*, is not allowed to contain *duplicated* alleles other than the 0 allele.

Fitness. The `Fitness` functions for ML and Minimum Cost clustering are obviously modeled on the corresponding objectives (27) and (31). For instance, in the ML case we have:

$$\text{Fitness}(\zeta) = \sum_{k:\zeta_k>0} \log\left(\hat{\tau}^{M}_{k\zeta_k} \Big/ \hat{\tau}^{U}_{k\zeta_k} \right) \tag{36}$$

where uninfluential constant terms appearing in (27) have been dropped.

Constraints. Even though only legal individuals are generated in the initial population, some *illegal* genotype may arise during evolution, due to `Reproduction`. In order to maintain a population of feasible candidate solution, these illegal genotypes are *repaired* by means of a purposefully designed operator. The `Repair` operator, $\texttt{Repair}:\zeta \to \zeta^*$, acts as a *stochastic* function mapping a genotype, ζ, into a *randomly repaired* version of it, ζ^*. If the ζ individual is legal, `Repair` leaves it unchanged. If, instead, ζ is illegal, `Repair` works as follows. Suppose ζ has ρ groups of duplicated non-zero alleles, with multiplicities n_r where $r = 1,\ldots,\rho$. For each group *r*, `Repair` first randomly selects inside the group just a single allele to be left unchanged, then it substitutes all the remaining $n_r - 1$ duplicates with the 0 allele.

Initial Population. As the search space of our EA is so huge, generating a good initial population is crucial. It is apparent that creating n_{ind} random individuals by *uniformly* sampling the search space would be a very poor choice. On the contrary, our algorithm samples more heavily those regions of the search space that are believed to be "more promising" on the basis of the already computed posterior probabilities (28). This is accomplished by the following Monte Carlo (MC) technique. First, the following posterior probabilities are computed for each record *k* in the smaller data set:

$$p^0_k = \Pr(Z_{ki} = 0 \;\forall i) = \prod_i \hat{\tau}^{U}_{ki} \tag{37}$$

$$p^j_k = \Pr((Z_{kj}=1) \;\; AND \;\; (Z_{ki}=0 \;\; \forall\, i \neq j)) = \hat{\tau}^{M}_{kj} \prod_{i \neq j} \hat{\tau}^{U}_{ki} \tag{38}$$

where $j = 1, 2, \ldots, n_{max}$. Value p^0_k is the posterior probability that the *k*-th record of the smaller data set does not match any record of the bigger, while p^j_k gives the posterior probability that the *k*-th record matches *only*[13] the *j*-th. The MC procedure generates *each* element ζ_k (for $k = 1, 2, \ldots, n_{min}$) of *each* genotype ζ of the initial population by sampling an allele value from $\{0, 1, \ldots, n_{max}\}$ with probability proportional to (37) and (38), i.e. $\Pr(\zeta_k = 0) \propto p^0_k$ and $\Pr(\zeta_k = j > 0) \propto p^j_k$. These MC generated genotypes are eventually processed by the The `Repair` operator, in order to warranty that the whole initial population is *legal*.

[13] Recall that the model in Section 5.1 does not encompass 1:1 constraints for the latent variable.

Selection. `Selection` is performed by means of a *rank-2 tournament*. Random pairs of individuals are formed. For each pair, the fitness of the individuals are compared. The fitter individual survives whereas the weaker dies and is dropped from the population, so as to make room for new individuals to be generated in the Reproduction phase. Notice that this `Selection` method is intrinsically *elitist*: the fittest individual of a generation surely survives and passes to the next generation.

Reproduction. Reproduction is performed as follows. Individuals that survived to `Selection` are randomly paired. Each pair generates two children. These children take the place of individuals that have been eliminated in the previous `Selection` phase. As a consequence the size of the population n_{ind} is kept *fixed* during the evolution. Children genotypes are obtained by merging those of the parents by means of *one-point crossover*. Call ζ^{p1} and ζ^{p2} the parents and ζ^{c1} and ζ^{c2} the children. A random cut point $cut \in \{1,\ldots,n_{min}-1\}$ is selected for the parents genotypes. Hence both ζ^{p1} and ζ^{p2} are cut into a *left* portion and a *right* portion. The first child receives the left portion from the first parent and the right from the second, i.e. $\zeta^{c1} = (\zeta_1^{p1},\ldots,\zeta_{cut}^{p1},\zeta_{cut+1}^{p2},\ldots,\zeta_{n_{min}}^{p2})$. The second child receives the left portion from the second parent and the right from the first, i.e. $\zeta^{c2} = (\zeta_1^{p2},\ldots,\zeta_{cut}^{p2},\zeta_{cut+1}^{p1},\ldots,\zeta_{n_{min}}^{p1})$. As there is no warranty that the generated children ζ^{c1} and ζ^{c2} are legal, they eventually undergo the `Repair` treatment before being plugged into the population.

Mutation. Each individual of the population has the same probability p_{muta} of undergoing `Mutation`. `Mutation` acts on a genotype ζ by affecting only a single allele. The outcome can be either that a nonzero allele is replaced by 0 (i.e. a declared Match is deleted from *Z*), or that a 0 allele is turned into a nonzero allele (i.e. a new declared Match is inserted into *Z*). The stochastic algorithm implementing `Mutation` exploits the *posterior estimate* of the number of Matches $\hat{n}_M = \sum_i \hat{z}_i$ obtained from (30) (or (32) for Minimum Cost). A random integer $p \in \{0,1,\ldots,n_{min}\}$ is drawn from a Binomial distribution with size n_{min} and success probability $\hat{n}_M / n_{min}$. If the genotype to mutate has more than *p* nonzero alleles, $\sum_k \text{sgn}(\zeta_k) > p$, then a random nonzero allele is replaced by 0. Otherwise, i.e. when $\sum_k \text{sgn}(\zeta_k) \leq p$, a random 0 allele is replaced by a nonzero one randomly selected from the set $\{1,\ldots,n_{max}\} \setminus \zeta$ (namely, by a new *legal* nonzero allele that did not already appear in ζ). Notice that, since the expected value of *p* is exactly $\hat{n}_M$, `Mutation` *tends on average* to delete declared Matches from candidate solution that contain *"too many"* of them, and conversely to add declared Matches to candidate solution that contain *"too few"* of them.

Termination Criterion. The Termination Criterion for the EA is two-fold. A first parameter, n_{gen}, controls the maximum number of generations that can be spent during evolution. If g denotes a generations counter, then the EA would stop as soon as $g > n_{\text{gen}}$. A second parameter, g_{stall}, gives the maximum number of generations that the EA is allowed to process without achieving a fitness improvement. If g' denotes the number of generations elapsed from the *last* fitness improvement, then the EA would stop as soon as $g' > g_{\text{stall}}$. The EA effectively stops as soon as either of the two conditions is verified.

Return Value. The return value of the EA is the genotype ζ^{Best} of the `Best Fit` individual found during evolution. This genotype is readily decoded into the corresponding phenotype matrix Z^{Best}, which in turn yields the *final* clustering result for the RL problem.

Memory Usage and Parameters Values. Storing a whole population of candidate solutions determines the EA memory overhead. As population size is kept fixed during evolution, if n denotes the size of the data sets to be matched then memory usage grows like $n_{\text{ind}} \times n$, i.e. almost linearly with n. Indeed, only a weak (less than linear) dependence of n_{ind} on n is expected. As a matter of fact, all the case studies listed in Section 6, despite their n values span over nearly an order of magnitude, have been carried out with the following default values for the EA parameters: $n_{\text{ind}} = 300$, $n_{\text{gen}} = 200$, $p_{\text{muta}} = 0.1$, $g_{\text{stall}} = 50$.

6. Experiments

Here we present an experimental evaluation of our mixture based suite of methods. Our focus will be on effectiveness; however, with respect to time complexity, we point out that our methods perform quadratically in the input data sets size (see equations (13), (17) for the fitting phase and (37), (38) for the clustering phase). Experiments have been carried out by using a comprehensive software system that implements all the methods proposed in the previous sections. We developed the system in the R programming language (R Development Core Team, 2009). All experiments have been run in an ordinary PC environment, equipped with: Windows XP 64 Operating System, 4 GB RAM, 2 GHz CPU.

We shall describe 9 RL instances involving 5 *very different* data sources. Indeed, a major aim of this section is to verify the *robustness* of our system against variations of the main characteristics of the RL problem. These include: data set size, Match rate (i.e. fraction of pairs that are Matches), type of records to be matched, number and discrimination power of variables used to compute distance measures, error rates affecting such variables, tendency of Unmatches to be similar even for clean data.

6.1 Experimental Setup

We first introduce the quality measures that we are going to use to assess the effectiveness of our RL system. We completely agree with (Christen et al., 2007) and hence avoid the Accuracy measure that, due to the huge class-skew inherent in RL problems, always gives a misleading impression of high effectiveness. On the contrary we choose to rely on traditional Precision (Prec) and Recall (Rec) measures. Moreover, whenever a single quality measure will be needed, we shall select the F-measure, $\mathrm{F} = 2/(\mathrm{Prec}^{-1} + \mathrm{Rec}^{-1})$. Notice that the F-measure is a *conservative* quality measure, as it can reach an high value only when *both* Precision and Recall are high.

A fundamental issue influencing our testing strategy concerns the *distance function* to be used in the RL process. As it should be clear from the previous sections, our mixture based approach does *not* rely on any restrictive assumption on the distance function (other than supposing it has unidimensional values). Therefore, our RL system can cope with *every* distance function (vectorial measures can easily be handled by averaging their components in a suitable way). Anyway, it is obvious that the choice of adopting a distance function rather than another for a specific RL task, can (and in general will) affect the

quality of the results. This is simply because, as a very rich literature in the RL field confirms (Elmagarmid et al., 2007), some distance functions are abler than others to capture some *specific aspects* of a given RL task. Nevertheless, since the comparison of alternative distance functions is completely beyond the scope of the present research, we shall use just a single distance for each RL instance. As a consequence, when reporting the quality of our results, the problem would arise of understanding how much of that quality actually depends on our methods, and how much, instead, on the adopted distance function. In other words, as the choice of the distance is just a free input for our system, we would like to build some kind of performance measure that is able to *factorize* the influence of the distance function. We developed such a "distance-independent" quality measure by exploiting what we call the 'Optimal Threshold Fully Supervised' classifier (OTFS).

The OTFS is a *theoretical* device. It is a 'Threshold'-based classifier in the sense that it classifies as Matches all the pairs with distance below a given threshold, and as Unmatches all the pairs above it. It is 'Fully Supervised' because it has full access to the true class labels of all pairs. It is an 'Optimal' classifier as, by knowing in advance the true results of the RL problem, it determines its classification threshold in such a way as to maximize the F-measure.[14] How to exploit the OTFS is easily understood. Indeed, given a quality metric Q (where Q $\in$ {Prec, Rec, F}), an approximately distance-independent measure of Q for our system can be computed as:

$$\Lambda_{\mathrm{Q}} = \mathrm{Q}_{\mathrm{SYS}}(dist)/\mathrm{Q}_{\mathrm{OTFS}}(dist) \tag{39}$$

where both classifiers, our system (SYS) and the OTFS, rely on the *same* distance function *dist*. We believe, in fact, that, even though both the numerator and the denominator in (39) depend on the choice of the distance, these dependencies will almost completely cancel out in the ratio.

6.2 Data Sources and RL Instances

Now we briefly describe the 9 proposed RL instances and the underlying 5 data sources, to which we shall refer as **Restaurants**, **Parks**, **Cens**, **Physics** and **PES**. Table 1 reports some basic information concerning these RL instances. With the only exception of **Cens**, all RL instances involve real-world data. All these problems are very hard, as indicated by (though not exclusively due to) their very low Match rates.

For all problems we choose the Levenshtein distance. When more than one matching variable is used, the following averaging procedure is adopted to obtain a scalar distance value. Call $\mathbf{d}^j = (d^j{}_1, \ldots, d^j{}_{n_P})$ the distance values measured with respect to the j-th matching variable on the n_P pairs,[15] and denote their mean and standard deviation with μ^j and σ^j. First, standardize these values and sum the standardized scores: $\mathbf{d}' = \sum_j [(\mathbf{d}^j - \mu^j)/\sigma^j]$. Then, simply normalize the obtained values in such a way that they fall inside the interval [0, 1], namely $\mathbf{d} = [\mathbf{d}' - \min(\mathbf{d}')]/[\max(\mathbf{d}') - \min(\mathbf{d}')]$.

[14] Notice that knowing in advance the true results of the OM problem is in general not sufficient for the OTFS to find a perfect classification threshold such that F = 1. Indeed, this is possible only if the histograms of the M and U distance distributions do not overlap at all.

[15] Whenever a variable had a missing value in one (or both) the records of a pair, we set the corresponding distance contribution to the blind average value 0.5.

Table 1 - Relevant Features of RL instances

RL Instance	Data Origin	Matching Variables	Data/Error Nature	Pairs (n_{min} x n_{max})	Matches	Match Rate
$Rest_1$	Riddle	name address city type	Real/Real	176,423 (331 x 533)	112	6.3E-4
$Rest_2$	Riddle	name	Real/Real	176,423 (331 x 533)	112	6.3E-4
Parks	SecondString	name	Real/Real	101,394 (258 x 393)	247	2.4E-3
Cens	SecondString	surname name midinit number street	Artificial/Artificial	176,008 (392 x 449)	327	1.9E-3
$Phys_1$	lanl.arXiv.org	title	Real/No	388,080 (588 x 660)	88	2.3E-4
$Phys_2$	lanl.arXiv.org	title	Real/Artificial	388,080 (588 x 660)	88	2.3E-4
PES_1	Istat	surname name sex birth.dd birth.mm birth.yyyy	Real/Real	1,033,272 (1,016 x 1,017)	984	9.5E-4
PES_2	Istat	surname name sex birth.dd birth.mm birth.yyyy	Real/Real	4,044,040 (2,002 x 2,020)	1,954	4.8E-4
PES_{full}	Istat	surname name sex birth.dd birth.mm birth.yyyy	Real/Real	32,876,434,096 (180,133 x 182,512)	172,621	5.3E-6

The **Restaurants** source, available at the RIDDLE[16] repository, contains restaurant records affected by real-world errors. It is used for two different RL tasks, **Rest$_1$** and **Rest$_2$**. They differ in the number of matching variables: **Rest$_1$** uses 4 variables, `name`, `address`, `city` and `type`, while **Rest$_2$** relies only on the `name` variable.

Both **Parks** and **Cens** sources are provided by the SECONDSTRING package.[17] Records in the **Parks** RL instance represent U.S. National Parks and the `name` of the park is used as the only matching variable. The **Cens** source, originally provided by William Winkler, contains synthetic census-like records; the corresponding RL instance relies on 5 matching variables: `surname`, `name`, `midinit`, `number` and `street`.

The **Physics** source refers to two partially overlapping selections of scientific papers in the field of high-energy physics.[18] These selections stem from two queries that have been intentionally designed to retrieve papers with very similar titles (even when papers are different). Accordingly, the **Physics** source is used for two different RL tasks, **Phys$_1$** and **Phys$_2$**, both constrained to adopt the `title` field as the only matching variable. The **Phys$_1$** task involves the original clean data, whereas artificially generated random errors have been introduced in **Phys$_2$**. This has been accomplished as follows. Each record from both data sets had a probability of 1/3 to be perturbed; for each selected record, first a number ν, ranging from 0 to the length of its `title` string λ, has been drawn from a Binomial distribution with size λ and success probability 1/6; then a random sample of ν characters drawn from the original `title` has been replaced by ν new random character values. The overall proportion of perturbed characters is about 6%.

PES$_1$, **PES$_2$** and **PES$_{full}$** involve data coming from the Post Enumeration Survey (PES) carried out by the Italian National Institute of Statistics to estimate the coverage rate of the 2001 population Census. Therefore, all the three RL instances deal with real-world data affected by real-world errors, including missing values. Each one of these RL tasks entails the matching of two lists of people, the first collected by the Census and the second by the PES; moreover for all the three RL tasks the same 6 matching variables are used: `surname`, `name`, `sex`, `birth.dd`, `birth.mm` and `birth.yyyy`. Both **PES$_1$** and **PES$_2$** tasks deal with a sample of enumeration areas belonging to the province of Rome, while **PES$_{full}$** faces the RL problem for the *whole* PES data.

As Table 1 shows clearly, **PES$_{full}$** represents a severe test-bed for assessing the practical feasibility of our methods when very large data sets are involved. Being the comparison-space so huge (about 33 *billions* of pairwise distances), a preliminary *blocking* step has been performed. The enumeration area code was selected as blocking variable. Since this variable was believed to be accurate (i.e. almost not affected by errors), the blocking step was expected to quickly filter-out pairs belonging, with high probability, to the U class. From a computational complexity point of view, the net result of the blocking phase was to transform the original, global RL task (which was not affordable) into a sequence of

[16] http://www.cs.utexas.edu/users/ml/riddle/index.html
[17] http://www.cs.utexas.edu/users/ml/riddle/data/secondstring.tar.gz
[18] http://xxx.lanl.gov/find/hep-ph, queries issued on 03/19/2009:
Keyword query 1 = abstract:(QCD and infrared)
Keyword query 2 = abstract:(QCD and confinement)

smaller, independent RL subtasks, one for each block. The overall number of processed blocks was 1,098. Correspondingly, the size of the comparison-space decreased from about 33 *billions* to about 86 *millions* pairs.

6.3 Results

For all the 9 instances described above, we ran our system choosing to solve the RL problem with the Maximum Likelihood objective. Though our system is able to deal also with the Minimum Cost objective, we did not consider this possibility in the experiments because: *i)* such a choice would have negatively affected the understanding and the comparability of our results; *ii)* the specification of misclassification costs is an application specific task.

The results are collectively shown in Table 2. It has to be stressed that, since $\mathbf{PES_{full}}$ is the only instance for which we used our RL system after a preliminary blocking step, and as every comparison-space reduction technique is a possible source of bias in the RL results, we excluded $\mathbf{PES_{full}}$ from the computation of the average performances reported in Table 2.

Table 2 - Precision, Recall and F-measure Results for the Proposed RL System (SYS)

	Precision			Recall			F-measure		
RL Instance	**OTFS**	**SYS**	Λ_{Prec}	**OTFS**	**SYS**	Λ_{Rec}	**OTFS**	**SYS**	Λ_F
$Rest_1$	0.941	0.933	**99.1%**	0.857	0.866	**101.0%**	0.897	0.898	**100.1%**
$Rest_2$	0.988	0.793	**80.2%**	0.759	0.786	**103.5%**	0.859	0.789	**91.9%**
Parks	0.934	0.971	**103.9%**	0.923	0.960	**103.9%**	0.929	0.965	**103.9%**
Cens	0.859	1.000	**116.4%**	0.911	0.982	**107.8%**	0.884	0.995	**112.6%**
$Phys_1$	1.000	0.854	**85.4%**	1.000	1.000	**100.0%**	1.000	0.921	**92.1%**
$Phys_2$	0.964	0.907	**94.1%**	0.909	1.000	**110.0%**	0.936	0.951	**101.7%**
PES_1	0.969	0.998	**102.9%**	0.997	0.996	**99.9%**	0.983	0.997	**101.4%**
PES_2	0.993	0.997	**100.4%**	0.984	0.996	**101.2%**	0.988	0.997	**100.8%**
PES_{full}	-	0.999	-	-	0.992	-	-	0.996	-
Average* Performance	0.956	0.932	**97.8%**	0.918	0.948	**103.4%**	0.934	0.939	**100.6%**

(*): Average values have been computed excluding the $\mathbf{PES_{full}}$ instance (see text).

A first look to the average Precision (0.932), Recall (0.948), and F-measure (0.939) performance immediately reveals the remarkable effectiveness of our system. Moreover, our systems exhibits also a very good robustness: Precision, Recall, and F-measures scores *never* significantly fall below 0.8, despite the addressed RL instances where deliberately selected to be very different.

Turning the attention to the "distance-independent" version of the three quality metrics (Λ columns, bold figures in the table), we observe that *i)* their average values are impressively high, Λ_{Prec} = 97.8%, Λ_{Rec} = 103.4%, Λ_F = 100.6%, and *ii)* all of them *never* fall below 80%. Again, these results strongly support the robustness of our methods. It is also interesting to note that the F-measure scores achieved by our system even outperform the OTFS classifier in 6 cases out of 8.

The **Rest$_2$** instance gives us the opportunity to compare our system with at least one previous proposal. Indeed, **Rest$_2$** exactly corresponds to one of the several RL instances considered in (Chaudhuri et al., 2005): same data sets (**Restaurants**), same matching variables (just `name`) and same distance function (Levenshtein distance). For this RL instance, authors of (Chaudhuri et al., 2005) present a Precision vs. Recall graph obtained when varying the parameters of their algorithms. Though the corresponding quality scores are not explicitly provided, it is possible to deduce from the aforementioned graph that, by fine-tuning parameters, their methods reach a maximum F-measure of about 0.54. We point out that our parameter-free system achieves an F-measure score of 0.789 for **Rest$_2$**, which means a relative gain in effectiveness of nearly 50%.

Moreover, our system also exhibits a very satisfactory behavior for the **PES$_{full}$** instance, from both the points of view of *effectiveness*[19] and *computational efficiency*. Indeed, on the one hand, the obtained Precision (0.999), Recall (0.992), and F-measure (0.996) scores are excellent. On the other hand, the run time performance of our system turned out to be very good. The overall execution time for processing about 86 millions of pairs, partitioned into 1,098 blocks, was 293 minutes (i.e. less than 5 hours) corresponding to an average processing time of about 2×10^{-4} seconds per pair.

7 Conclusions

In this paper, we presented a novel approach to the RL problem based on mixture models. Several original contributions enable our methods to be at the same time effective and fully automated. We validated our suite of methods by testing its Precision, Recall and F-measure scores on real data sets, obtaining excellent results. Our extensive experimental study, which deliberately involved very different RL instances, also showed the remarkable robustness of our methods.

[19] For the **PES$_{full}$** instance, running the OTFS was computationally unfeasible (whence the lacking scores in Table 2). This is because the OTFS cannot be used after a blocking step: indeed, by definition, it has to process the distance distribution of all the pairs as a whole.

References

Armstrong, J. and Mayda, J., "Estimation of Record Linkage Models Using Dependent Data", *Proceedings of the Survey Research Methods Section*, American Statistical Association, 1992.

Belin, T. and Rubin, D., "A Method for Calibrating False-Match Rates in Record Linkage", *Journal of the American Statistical Association*, 90, 1995.

Bender, C. and Orszag, S., *Advanced Mathematical Methods for Scientists and Engineers: Asymptotic methods and perturbation theory*, Springer, New York, 1999.

Chaudhuri, S. and Ganti, V. and Motwani, R., "Robust Identification of Fuzzy Duplicates", *Proceedings of the International Conference on Data Engineering*, 2005.

Christen, P., "A two-step classification approach to unsupervised record linkage", *Proceedings of the Australasian Data Mining Conference*, 2007.

Christen, P. and Goiser, K., "Quality and complexity measures for data linkage and deduplication", In F. Guillet and H. Hamilton, editors, *Quality Measures in Data Mining*, Springer Studies in Computational Intelligence, 2007.

Dempster, A. and Laird, N. and Rubin, D., "Maximum-likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, *SERIES B*, 39(1), 1977.

Duda, R. and Hart, P. and Stork, D., *Pattern Classification*, John Wiley & Sons, 2000.

Elmagarmid, A. and Ipeirotis, P. and Verykios, V., "Duplicate Record Detection: A Survey", *IEEE Transactions on Knowledge and Data Engineering*, 19(1), 2007.

Fellegi, I. and Sunter, A., "A Theory for Record Linkage", *Journal of the American Statistical Association*, 64, 1969.

Fortini, M. and Liseo, B. and Nuccitelli, A. and Scanu, M., "On Bayesian Record Linkage", *Research in Official Statistics*, 4(1), 2001.

Guha, S. and Koudas, N. and Marathe, A. and Srivastava, D., "Merging the Results of Approximate Match Operations", *Proceedings of the International Conference on Very Large Databases*, 2004.

Jaro, M., "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida", *Journal of the American Statistical Association*, 84, 1989.

Larsen, M., "Modeling issues and the use of experience in record linkage", *Proceedings of the Federal Committee on Statistical Methodology*, Record Linkage Workshop, 1997.

Larsen, M. and Rubin, D., "Iterative Automated Record Linkage Using Mixture Models", *Journal of the American Statistical Association*, 96(453), 2001.

Larsen, M., "Hierarchical Bayesian Record Linkage Theory", *Proceedings of the Survey Research Methods Section*, American Statistical Association, 2005.

McLachlan, G. and Basford, K., *Mixture Models: Inference and Applications to Clustering*, M. Dekker, New York, 1988.

McLachlan, G. and Peel, D., *Finite Mixture Models*, John Wiley & Sons, 2000.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, 1996.

Ospina, R. and Ferrari, S. L. P., "Inflated Beta Distributions". *Statistical Papers*, 51-1, Springer, Berlin, 2010.

R Development Core Team (2009), "R: A Language and Environment for Statistical Computing". *R Foundation for Statistical Computing*, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Tejada, S. and Knoblock, C. and Minton, S., "Learning object identification rules for information integration", *Information Systems*, 26(8), 2001.

Verykios, V. and Elmagarmid, A. and Houstis, E., "Automating the approximate record-matching process", *Journal of Information Sciences*, 126(1-4), 2000.

Verykios, V. and Moustakides, G. and Elfeky, M., "A Bayesian Decision Model for Cost Optimal Record Matching", *VLDB Journal*, 12(1), 2003.

Winkler, W., "Improved decision rules in the Fellegi-Sunter model of record linkage", *Proceedings of the Survey Research Methods Section*, American Statistical Association, 1993.

Winkler, W., "Advanced Methods for Record Linkage", *Proceedings of the Survey Research Methods Section*, American Statistical Association, 1994.