Automatic Training Example Selection for Scalable Unsupervised Record Linkage

Peter Christen

Department of Computer Science, The Australian National University
Canberra ACT 0200, Australia
peter.christen@anu.edu.au

Abstract. Linking records from two or more databases is becoming increasingly important in the data preparation step of many data mining projects, as linked data can enable analysts to conduct studies that are not feasible otherwise, or that would require expensive and time-consuming collection of specific data. The aim of such linkages is to match all records that refer to the same entity. One of the main challenges in record linkage is the accurate classification of record pairs into matches and non-matches. With traditional techniques, classification thresholds have to be set either manually or using an EM-based approach. Many modern classification techniques, on the other hand, are based on supervised machine learning and thus require training data, which is often not available in real world situations. A novel two-step approach to unsupervised record pair classification is presented in this paper. In the first step, training examples are selected automatically, and in the second step these examples are used to train a binary classifier. An experimental evaluation shows that this approach can outperform k-means clustering and can also be much faster than other classification techniques.

Keywords: data linkage, entity resolution, clustering, support vector machines, data mining preprocessing.

1 Introduction

With massive amounts of data being collected by many businesses, government agencies and research projects, techniques that enable efficient and automatic sharing of large databases between organisations are of increasing importance in many data mining projects. Data from various sources often has to be linked and aggregated in order to improve data quality and integrity, or to enrich existing data with additional information [16]. The aim of such linkages is to match all records that refer to the same entity, for example a customer, a patient, or a business. A related task is finding duplicate records that refer to the same entity within one database, as such duplicates can significantly affect data quality.

Record linkage has traditionally been employed in the health sector for epidemiological studies [11] and within statistical agencies for linking census data [17]. Today, businesses increasingly use deduplication and linkage techniques to improve the quality of their data, for example when compiling mailing

lists or when linking data within collaborative e-Commerce projects. Within government agencies, such as taxation offices and departments of social security, record linkage is used to identify people who register for assistance multiple times or who work and collect unemployment benefits. Another area where record linkage techniques are increasingly being used is fraud, crime and terror detection. Security agencies often require fast access to files of a particular individual in order to solve crimes or to prevent terror through early intervention.

Linking entities is often challenged by the lack of unique entity identifiers, and thus more sophisticated linkage techniques, using the available record attributes, are required [8, 17]. The naive approach for linking two databases, to compare each record in one database with all records in another database, is of quadratic complexity. Because the performance bottleneck in a record linkage system is usually the computationally expensive comparison of fields (or attributes) between pairs of records [8], blocking, filtering or indexing techniques are normally employed to reduce the large amount of potential record pair comparisons [1]. These techniques group records into blocks according to some criteria (such as having the same value in a 'postcode' attribute). Candidate record pairs are then generated only from the records within the same block. Assuming there are no duplicate records in the databases to be linked, then the majority of candidate pairs will likely be non-matches, as the maximum possible number of true matches corresponds to the number of records in the smaller of the databases. Classifying record pairs is therefore often an imbalanced classification problem.

Candidate record pairs are compared using various similarity functions applied to selected record attributes. These functions can be as simple as an exact string or a numerical comparison, can take variations into account [3], or they can be specialised for attributes that contain dates, times, or even geographic locations. Each comparison returns a numerical similarity value (called *matching weight*), often in normalised form between 0.0 (for totally different attribute values) and 1.0 (for exactly matching values). A *weight vector* is formed for each compared record pair (as shown in Fig. 1) containing all matching weights calculated when comparing the pair's attribute values. Using these weight vectors, candidate record pairs are then classified into *matches*, *non-matches*, and *possible matches*, depending upon the decision model used [8].

It can generally be assumed that a record pair that has the same or very similar values in all its record attributes will likely refer to the same entity, as it is very unlikely that two entities have very similar or even the same values in all their attributes. The matching weights in the vector calculated when comparing such a pair will be 1 (or close to 1) in all vector elements. On the other hand, weight vectors that contain matching weights of only 0 (or values close to 0) in all vector elements were with high likelihood calculated when two different entities were compared, as it is highly unlikely that two records that refer to the same entity have different values in all their record attributes.

Based on these observations, it is usually easy to accurately classify a record pair as a match when its corresponding weight vector contains mainly matching weights close to or equal to 1, and as a non-match when its matching weights

	Name		Address		
R1:	Christine	Smith	42	Main	Street
R2:	Christina	Smith	42	Main	St
R3:	Bob	O'Brian	11	Smith	Rd
R4:	Robert	Bryce	12	Smythe	Road

WV(R1,R2): [0.9, 1.0, 1.0, 1.0, 0.9]
WV(R1,R3): [0.0, 0.0, 0.0, 0.0, 0.0]
WV(R1,R4): [0.0, 0.0, 0.5, 0.0, 0.0]
WV(R2,R3): [0.0, 0.0, 0.0, 0.0, 0.0]
WV(R2,R4): [0.0, 0.0, 0.5, 0.0, 0.0]
WV(R3,R4): [0.7, 0.3, 0.5, 0.7, 0.9]

Fig. 1. The left side shows four example records and the right side the corresponding weight vectors resulting from their comparisons (based on Fig. 2 from [6]).

are mainly close to or equal to 0. It is however much more difficult to correctly classify a pair that contains some attribute values that are similar while others are not (i.e. its weight vector contains some matching weights close to 1 and others close to 0). In the examples shown in Fig. 1, records *R1* and *R2* are very similar, with only small differences in their given name and street type values (and thus have matching weights close to or equal to 1), and thus very likely refer to the same person. Records *R3* and *R4*, on the other hand, are more different from each other, and it is not obvious if they refer to the same person.

It follows that it is possible to automatically select in a first step weight vectors as match training examples of good quality that very likely were generated when two records that refer to the same entity were compared, and similarly to select non-match training examples from the many weight vectors that were generated when records that refer to two different entities were compared. For example, of the weight vectors shown in Fig. 1, *WV(R1,R2)* can be selected as a match training example, and *WV(R1,R3)*, *WV(R2,R3)*, *WV(R1,R4)* and *WV(R2,R4)* as non-match examples. These training examples can then be used in a second step to train a classifier for classification of all weight vectors.

This two-step approach to unsupervised classification of record pairs has first been proposed by the author in [6], with initial experiments indicating its feasibility. The contributions of this paper are the investigation and evaluation of a potential improvement to the basic approach, namely to randomly include additional weight vectors for training; and an evaluation of the scalability of the approach. First, in Sect. 2, related work is summarised. Section 3 then presents the proposed approach in detail, and in Sect. 4 both its accuracy and scalability are evaluated. The paper is concluded by an outlook to future work in Sect. 5.

2 Related Work

In recent years, various techniques have been explored in record linkage for the classification of record pairs. The traditional probabilistic approach has been improved by applying the expectation-maximisation (EM) algorithm for better parameter estimation [17]. Various supervised learning approaches have been investigated, among them decision trees [9, 15] and support vector machines [13]. Another approach is to learn string similarity measures, such as the costs for

edit distance operations [3], in order to adapt similarity calculations to a certain domain. While supervised techniques normally achieve better linkage quality than unsupervised approaches, their major drawback is the lack of training data (record pairs with known true match and non-match status) in many real world situations. Manual preparation of training examples is time consuming, cumbersome and expensive. Active learning is an approach that aims to overcome this problem [15]. Only the record pairs most difficult to classify automatically are provided for manual classification, and subsequent re-training of a classifier.

Three classification approaches were presented in [9]: decision tree induction; unsupervised k-means clustering with a cluster each for matches, possible matches and non-matches; and a hybrid approach that first clusters a sub-set of all weight vectors (again into three clusters), and then uses the match and non-match clusters for decision tree induction learning. The supervised and hybrid approaches both outperformed clustering. K-means clustering was also used in [10] to cluster weight vectors into matches and non-matches, with the possibility to select the weight vectors in the region half-way between the match and non-match centroids as possible matches, using a selectable threshold.

In recent years, unsupervised techniques have been developed for collective entity resolution of relational data [2], i.e. data that contains relational information linking different types of entities (like publications, authors, and conferences). For such data, relational entity resolution outperforms techniques that only use pairwise similarities between records. In the real world, there are however large amounts of data that do not allow relational entity resolution. The aim of this paper is to improve unsupervised classification for non-relational data.

Methods similar to the approach presented here have recently been developed for text and Web page classification [12, 14, 18], where besides many unlabeled documents often only a small number of positive labeled training examples is available. In such situations, the aim is to learn a binary classifier from positive and unlabeled examples. PEBL [18] iteratively trains a support vector machine using the positive and the strongest negative examples (i.e. the documents furthest away from the decision boundary), while the S-EM [12] approach includes 'spy' documents, positive labeled examples, into the set of unlabeled documents to get a more realistic model of their distribution to be used in the EM algorithm. This is similar to the idea of randomly including additional weight vectors into the training sets as presented and evaluated in this paper. The EM algorithm and a Naïve Bayes classifier have been combined in [14] for the situation where only a small number of all available documents are labeled as positive or negative training examples. The training is initially based on only the labeled examples, but then iteratively refined using unlabeled documents as well.

3 Two-step Classification

In the first step of the proposed classification approach, weight vectors are selected as training examples that with high likelihood correspond to true matches and true non-matches. For match training examples, weight vectors containing

only exact or high similarity values are selected, while for non-match examples vectors containing only low similarity or total dissimilarity values are chosen. In the second step, these training examples are used to train a classifier, which is then employed to classify all weight vectors into matches and non-matches.

3.1 Training Example Selection

There are two different approaches on how to select training examples: threshold or nearest based [6]. In the first approach, weight vectors that have all their vector elements within a certain distance to the exact similarity or total dissimilarity values, respectively, will be selected. For example, using the weight vectors from Fig. 1 and a distance threshold of 0.2, only vector *WV(R1,R2)* will be selected as match training example, and *WV(R1,R3)* and *WV(R2,R3)* as non-match training examples. The remaining three weight vectors will not be selected as at least one of their vector elements is larger than the 0.2 distance threshold.

The second approach is to sort weight vectors according to their distances from the vectors containing only exact similarities and only total dissimilarities, respectively, and to then select the respectively nearest vectors. In Fig. 1, vector *WV(R1,R2)* is closest to the exact similarities vector, followed by *WV(R3,R4)*. Vectors *WV(R1,R3)* and *WV(R2,R3)* only contain total dissimilarity values, and *WV(R1,R4)* and *WV(R2,R4)* are the vectors next closest to them.

Both approaches are presented more formally below. First, the notation used in this paper is provided. It is assumed that candidate record pairs are compared using d comparison functions (with $d \geq 1$), resulting in a set $\mathbf{W}$ of weight vectors $\mathbf{w}_i$ ($1 \leq i \leq |\mathbf{W}|$) of length d containing matching weights (similarity values), with $|\cdot|$ denoting the number of elements in a set. It is also assumed that all comparison functions return normalised similarity values between 0 (total dissimilarity) and 1 (exact similarity), i.e. $0.0 \leq \mathbf{w}_i[j] \leq 1.0, 1 \leq j \leq d, \forall \mathbf{w}_i \in \mathbf{W}$. The weight vector containing exact similarities in all vector elements (i.e. corresponding to an exact match) is denoted by $\mathbf{m}$ (with $\mathbf{m}[j] = 1.0, 1 \leq j \leq d$), and the vector with only dissimilarities by $\mathbf{n}$ (with $\mathbf{n}[j] = 0.0, 1 \leq j \leq d$).

The aim of the training example selection step is to chose weight vectors from $\mathbf{W}$ and insert them into two sets: the match training examples, $\mathbf{W}_M$, and the non-match training example, $\mathbf{W}_N$, such that weight vectors in $\mathbf{W}_M$ with very high likelihood correspond to true matches, and weight vectors in $\mathbf{W}_N$ to true non-matches. In general, not all weight vectors from $\mathbf{W}$ will be selected for training, thus it is likely that $(|\mathbf{W}_M| + |\mathbf{W}_N|) < |\mathbf{W}|$ holds.

Threshold-based Selection One distance threshold for matches, t_M, and one for non-matches, t_N (with $0.0 < t_M, t_N < 1.0$), are used in this approach to select weight vectors that have all their similarity values either within t_M of the exact match value $\mathbf{m}$, or within t_N of the total dissimilarity value $\mathbf{n}$. Formally, weight vectors from $\mathbf{W}$ will be inserted into $\mathbf{W}_M$ and $\mathbf{W}_N$, according to:

$$\mathbf{W}_M = \{\mathbf{w}_i \in \mathbf{W} : (\mathbf{m}[j] - \mathbf{w}_i[j]) \leq t_M,\ 1 \leq j \leq d\},$$
$$\mathbf{W}_N = \{\mathbf{w}_i \in \mathbf{W} : (\mathbf{n}[j] + \mathbf{w}_i[j]) \leq t_N,\ 1 \leq j \leq d\}.$$

In a situation where $(t_M + t_N) \geq 1.0$ it is possible that weight vectors could be included into both $\mathbf{W}_M$ and $\mathbf{W}_N$. If this happens, these vectors will be removed from both sets, as they cannot be used as training example for both matches and non-matches. This would for example happen if $t_M = t_N = 0.6$ for a vector with similarity values 0.5 in all its elements, i.e. $\mathbf{w}_i[j] = 0.5, 1 \leq j \leq d$.

Nearest-based Selection In this approach the x_M weight vectors closest to $\mathbf{m}$ are selected into $\mathbf{W}_M$, and the x_N weight vectors closest to $\mathbf{n}$ are selected into $\mathbf{W}_N$. Both $x_M > 0$ and $x_N > 0$ must hold. If, for example, Manhattan distance is used to calculate the distance between two weight vectors $\mathbf{w}_i$ and $\mathbf{w}_k$, then the training example sets $\mathbf{W}_M$ and $\mathbf{W}_N$ are formed according to:

$$\mathbf{W}_M = \{\mathbf{w}_i \in \mathbf{W}, \mathbf{w}_k \notin \mathbf{W}_M : dist(\mathbf{m}, \mathbf{w}_i) < dist(\mathbf{m}, \mathbf{w}_k)\},$$
$$\mathbf{W}_N = \{\mathbf{w}_i \in \mathbf{W}, \mathbf{w}_k \notin \mathbf{W}_N : dist(\mathbf{w}_i, \mathbf{n}) < dist(\mathbf{w}_k, \mathbf{n})\},$$

with $x_M = |\mathbf{W}_M|$, $x_N = |\mathbf{W}_N|$, and $dist(\mathbf{w}_i, \mathbf{w}_k) = \sum_{j=1}^{d} |\mathbf{w}_i[j] - \mathbf{w}_k[j]|$. Other distance functions, such as Euclidean distance, can be used alternatively.

In most linkage situations, there will be a large number of record pairs that contain only totally different attribute values, resulting in weight vectors with only dissimilarity values (such as *WV(R1,R3)* and *WV(R2,R3)* in Fig. 1). In the worst case, all weight vectors selected into $\mathbf{W}_N$ could be equal to $\mathbf{n}$. Similarly, a number of record pairs might be exact matches, resulting in several weight vectors being equal to $\mathbf{m}$. Such a situation would not be very useful for training a classifier. Assuring that only the x_M and x_N unique nearest vectors will be selected into $\mathbf{W}_M$ and $\mathbf{W}_N$ should therefore improve the classification accuracy. Thus, the two variations of the nearest based approach are to either select the x_M and x_N nearest vectors, regardless if some of them contain the same values in all of their vector elements; or to select the x_M and x_N *unique* nearest vectors.

A question that arises is how to choose the numbers x_M and x_N of nearest weight vectors to select. One option is to select the same number into $\mathbf{W}_M$ and $\mathbf{W}_N$, so that $x_M = x_N$, leading to a balanced classification problem. Given that the number of non-matches in $\mathbf{W}$ is often much larger than the number of matches [8], alternatively selecting imbalanced numbers of training examples should result in more realistic training data. The danger with balanced training set selection is that weight vectors that more likely don't refer to true matches might be selected into $\mathbf{W}_M$. An estimation of the ratio r of matches to non-matches can be calculated using the number of records in the two data sets to be linked, $\mathbf{A}$ and $\mathbf{B}$, and the number of weight vectors $|\mathbf{W}|$: $r = min(|\mathbf{A}|, |\mathbf{B}|)/(|\mathbf{W}| - min(|\mathbf{A}|, |\mathbf{B}|))$.

3.2 Random Inclusion of Additional Training Examples

The training data automatically selected in the first step will likely be linearly separable, because the two training sets, $\mathbf{W}_M$ and $\mathbf{W}_N$, only contain weight vectors that are either close to the exact match vector $\mathbf{m}$ or close to the total

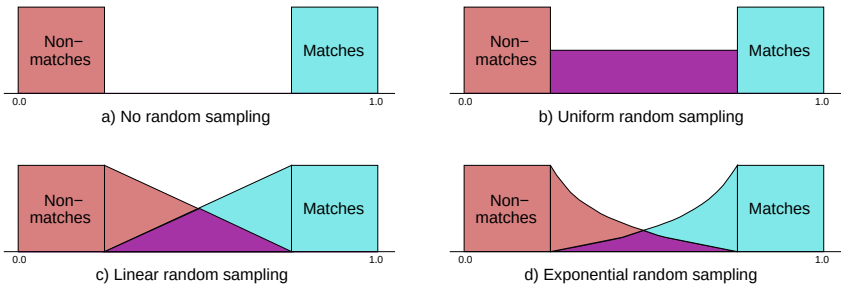

Fig. 2. Possible methods for random sampling of additional weight vectors (assumed to be 1-dimensional vectors).

dissimilarity vector $\mathbf{n}$, and also because usually not all weight vectors from $\mathbf{W}$ will be selected for training. This will likely result in a 'gap' between the training sets, as illustrated in Fig. 2 a). Similar to the inclusion of 'spy' documents for semi-supervised text classification [12], adding a small number of randomly selected weight vectors from this 'gap' into the training example sets should help to improve classification accuracy, because the training sets will then have a more realistic distribution of weight vectors.

The random sampling of weight vectors should be done in such a way that vectors closer to $\mathbf{m}$ are more likely included into $\mathbf{W}_M$, while vectors closer to $\mathbf{n}$ should more likely be selected for $\mathbf{W}_N$. Besides no random sampling, the three different sampling methods illustrated in parts b) to d) of Fig. 2 are to use either uniform sampling, or a linear or exponential mapping function to randomly sample weight vectors. Intuitively, exponential should outperform linear sampling, which in turn should be better than uniform and no sampling.

3.3 Weight Vector Classification

In the second step of the proposed record pair classification approach, the training sets $\mathbf{W}_M$ and $\mathbf{W}_N$, as generated in the first step, will be used to train a binary classifier. Once trained, this classifier is then employed to classify all weight vectors in $\mathbf{W}$. In the experiments presented below, a support vector machine (SVM) classifier [4] will be evaluated, because this technique can handle high-dimensional data and is known to be robust to noisy data.

While training a classifier once to classify all weight vectors in $\mathbf{W}$ is the basic approach in this second step, related work in Web page classification has shown that an iterative approach can improve classification accuracy [18]. The idea is to train a classifier first using only the training sets $\mathbf{W}_M$ and $\mathbf{W}_N$, and to then iteratively include the strongest classified matches and non-matches, i.e. the weight vectors furthest away from the decision boundary, into the training sets. The improved training sets are then used in the following iteration, and this process is repeated until a stopping criteria is fulfilled. This approach is currently being implemented and results will be reported in the near future.

Table 1. Data sets used in experiments. See Sect. 4.1 for more details.

Data set	Number of records	Task	Pairs completeness	Reduction ratio	Number of weight vectors
Census	449 + 392	Link	1.000	0.988	2,093
Restaurant	864	Dedup	1.000	0.713	106,875
Cora	1,295	Dedup	0.924	0.793	173,769
DS-Gen-A	1,000	Dedup	0.957	0.995	2,475
DS-Gen-B	2,500	Dedup	0.940	0.997	9,878
DS-Gen-C	5,000	Dedup	0.953	0.997	35,491
DS-Gen-D	10,000	Dedup	0.948	0.997	132,532

4 Experimental Evaluation

The proposed two-step approach to automatic record pair classification will be compared with three other classification methods. The first is an 'optimal threshold' classifier that has access to the true match status of all weight vectors in $\mathbf{W}$ and can thus find an optimal classification threshold. For each weight vector, all its vector elements are summed into one matching weight, and the classification threshold that minimises both false matches and false non-matches is calculated over all summed matching weights. The second classification method is a supervised SVM which also has access to the true match status of all weight vectors. Nine SVM variations were evaluated (three kernels: linear, polynomial and RBF; and three values for the cost parameter, C [4]: $0.1, 1, 10$). The third method is k-means clustering that has previously been used for record pair classification [9, 10]. Weight vectors were grouped into a match and a non-match cluster, with the initial centroids set to $\mathbf{m}$ and $\mathbf{n}$, respectively. Three distance measures (Manhattan, Euclidean and L_{inf}) were evaluated. All experiments were conducted using 10-fold cross validation.

The discussed techniques were implemented in the *Febrl* [7] open source record linkage system, which is written in the Python programming language. The *libsvm* library was used for the SVM classifier [4]. All experiments were conducted on a Dell Optiplex GX280 with an Intel Pentium 3 GHz CPU and 2 GBytes of main memory, running Linux 2.6.20 and using Python 2.5.1.

4.1 Data Sets and Linkage Setup

The proposed approach was evaluated using the data sets summarised in Table 1. Three real data sets from the *SecondString* toolkit[1] were used, and artificial data sets of various sizes containing names and addresses were created randomly using the *Febrl* data set generator [5]. This data was created based on real-world frequency tables, with duplicates generated randomly using modifications like

[1] `http://secondstring.sourceforge.net`

Table 2. Quality of training example selection, adapted from [6]. Each pair of result values shows the quality of $\mathbf{W}_M/\mathbf{W}_N$ as percentages of correctly selected training examples. '–' denotes an empty training set. Nearest-based selection was imbalanced. 'NU' stands for non-unique selection of weight vectors, and 'U' for unique selection.

Data sets	Thresholds			Nearest NU		Nearest U	
	0.3	0.5	0.7	1%	10%	1%	10%
Census	100/–	96.2/100	73.4/100	100/100	100/100	100/100	100/100
Restaurant	98.5/–	4.5/100	0.19/100	100/100	90.8/100	100/100	58.6/100
Cora	99.7/100	99.9/97.0	99.5/99.0	100/96.8	100/97.6	100/98.2	99.2/98.4
DS-Gen-A	100/100	100/100	100/99.0	100/100	100/95.5	100/100	100/95.5
DS-Gen-B	100/100	100/100	99.8/99.4	100/99.0	100/98.3	100/99.0	100/98.2
DS-Gen-C	100/100	100/100	98.0/99.7	100/100	100/99.5	100/99.7	100/99.6
DS-Gen-D	100/99.7	100/100	95.5/99.9	100/99.9	100/99.7	100/99.8	100/99.7

character inserts, deletes or substitutions; and swapping, removing, inserting, splitting or merging of words. All artificial data sets generated for the experiments in this paper contained 60% original and 40% duplicate records.

Standard blocking [1] was applied to reduce the number of record pair comparisons, and the Winkler approximate string comparison technique [16] was used for comparing name and address values. Additionally, character difference comparison was used on attributes such as postcode, year, or street number.

In Table 1, the quality and complexity of the compared record pairs is shown using the measures *pairs completeness* (the number of true matched record pairs generated by blocking divided by the total number of true matched pairs) and *reduction ratio* (number of record pairs generated by blocking divided by all possible record pairs) [8, 9]. Accuracy, as commonly used for measuring classifier performance, is not suitable for assessing the quality of the classified record pairs due to the normally imbalanced distribution of matches and non-matches in the weight vector set $\mathbf{W}$ [8]. The F-measure, $F = 2PR/(P+R)$, the harmonic mean of precision ($P = TP/(TP + FP)$) and recall ($R = TP/(TP + FN)$) is used instead, with TP and FP being the number of true and false positives (matches), and TN and FN the number of true and false negatives (non-matches).

The quality of the training example sets generated in step one of the proposed approach, as shown in Table 2, is calculated as the percentage of correctly selected weight vectors in the training example sets, i.e. (|true matches in $\mathbf{W}_M$|/|$\mathbf{W}_M$|) and (|true non-matches in $\mathbf{W}_N$|/|$\mathbf{W}_N$|).

4.2 Training Example Quality

As can be seen from Table 2, the quality of the training example sets $\mathbf{W}_M$ and $\mathbf{W}_N$ is very good in most cases. For the threshold based approach, a threshold of 0.5 achieved the best results, while a lower threshold can produce empty

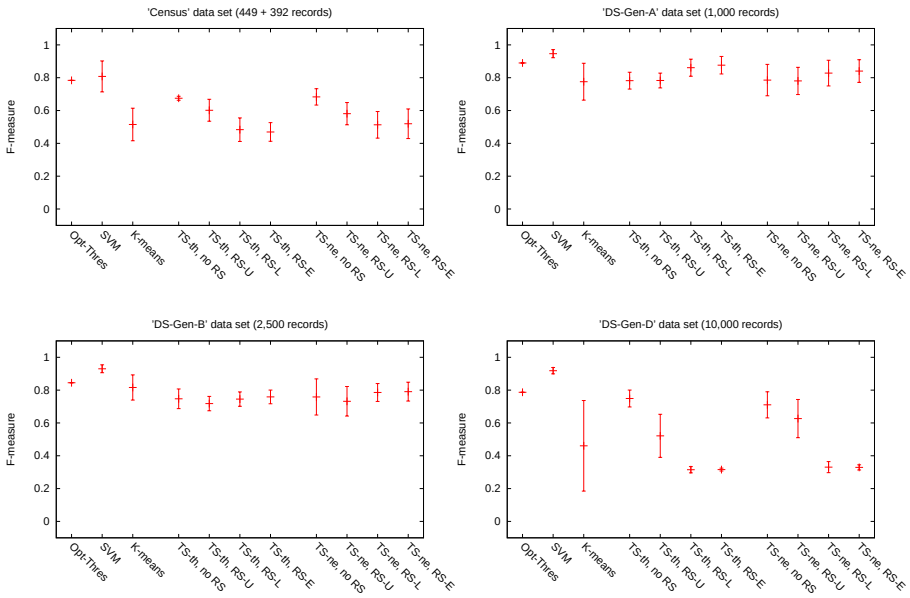

Fig. 3. F-measure results (averages and standard deviations). 'RS' stands for random selection, 'U' for uniform, 'L' for linear and 'E' for exponential, while 'ne' stands for nearest and 'th' for threshold based selection. Nearest-based selection was imbalanced.

training sets (denoted by '–'), if all weight vectors have at least one matching weight with a similarity value above the selected threshold (i.e. the two records in the corresponding pair had at least one attribute with a similar value).

Nearest-based selection overcomes this problem, and generally results in very good quality training sets. With balanced nearest selection [6] (not shown here) too many weight vectors are included into the match training set $\mathbf{W}_M$, significantly reducing its quality in certain cases. There is no significant difference between unique and non-unique training example selection.

4.3 Classification Performance

Figure 3 show the F-measure results for four data sets (due to space limitations not all results can be shown) and over the parameter settings described in Sect. 4 (nine variations for SVM and two-step, and three for k-means). The four random selection methods described in Sect. 3.2 are shown for the two-step classifier approach. As can be seen, both supervised classifiers (optimal threshold and SVM) achieved the highest linkage quality. With the exception of the 'DS-Gen-B' data set, the two-step classification approach outperformed k-means, achieving significantly better results for the 'Census' and 'DS-Gen-D' data sets. While for data sets 'DS-Gen-A' and 'DS-Gen-B' random selection using the linear or exponential methods achieves slightly better classification results, for the

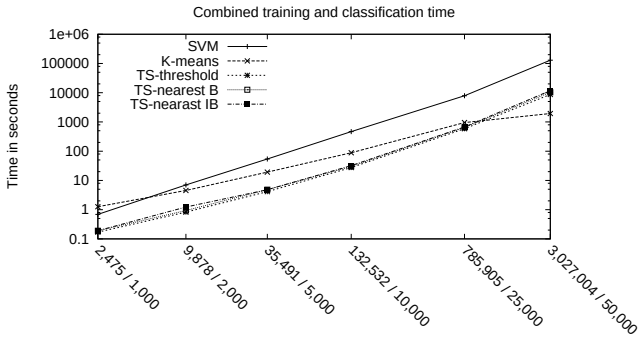

Fig. 4. Timing results for synthetic data sets of various sizes.

other data sets all random inclusion methods worsen the quality of the training sets and result in significantly reduced classification performance. This indicates that, unlike the random inclusion of 'spy' documents for semi-supervised text classification [12], inclusion of additional randomly selected weight vectors is not a technique suitable for record pair classification.

4.4 Timing and Scalability

In order to evaluate the scalability of the proposed record pair classification approach, a series of experiments with synthetic data sets of increasing sizes were conducted. Euclidean distance was used for k-means clustering, while a RBF kernel was selected for the SVM and two-step classifiers. Nearest-based training example selection was used with 5% of weight vectors included into each training set (no random inclusion of additional vectors). As Fig. 4 shows, the SVM and two-step approaches are of similar complexity, with the latter being one magnitude faster, as only 10% of all weight vectors were used for training.

5 Conclusions and Future Work

A novel two-step approach to record pair classification has been presented and evaluated in this paper. In a first step high-quality training examples are selected automatically, to be used in a second step to train a binary classifier. Together, these two steps allow unsupervised classification of record pairs with often better linkage quality than k-means clustering. Contrary to expectations, the inclusion of randomly selected additional weight vectors did not result in increased classification performance. Timing experiments showed that the proposed approach can be a magnitude faster than a supervised SVM classifier.

Future work includes the implementation and evaluation of an approach that iteratively refines the training example sets by including the strongest classified matches and non-matches, followed by re-training of the classifier, similar to the PEBL approach developed for text and Web page classification [18].

Acknowledgements

This work is supported by an Australian Research Council (ARC) Linkage Grant LP0453463 and partially funded by the New South Wales Department of Health. The author would like to thank Paul Thomas for proof-reading.

References

1. R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *ACM KDD'03 workshop on Data Cleaning, Record Linkage and Object Consolidation*, pages 25–27, Washington DC, 2003.
2. I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.
3. M. Bilenko and R.J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *ACM KDD'03*, pages 39–48, Washington DC, 2003.
4. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. Manual, Department of Computer Science, National Taiwan University, 2001. Software available at: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
5. P. Christen. Probabilistic data generation for deduplication and data linkage. In *IDEAL'05, Springer LNCS 3578*, pages 109–116, Brisbane, 2005.
6. P. Christen. A two-step classification approach to unsupervised record linkage. In *AusDM'07, CRPIT. 70*, Gold Coast, Australia, 2007.
7. P. Christen, T. Churches, and M. Hegland. Febrl – A parallel open source data linkage system. In *PAKDD'04, Springer LNAI 3056*, pages 638–647, Sydney, 2004.
8. P. Christen and K. Goiser. Quality and complexity measures for data linkage and deduplication. In F. Guillet and H. Hamilton, editors, *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 127–151. Springer, 2007.
9. M.G. Elfeky, V.S. Verykios, and A.K. Elmagarmid. TAILOR: A record linkage toolbox. In *ICDE'02*, pages 17–28, San Jose, 2002.
10. L. Gu and R. Baxter. Decision models for record linkage. In *Selected Papers from AusDM, Springer LNCS 3755*, pages 146–160, 2006.
11. C.W. Kelman, J. Bass, and D. Holman. Research use of linked health data – A best practice protocol. *Aust NZ Journal of Public Health*, 26:251–255, 2002.
12. B. Liu, W.S. Lee, P.S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML'02*, pages 387–394, Sydney, Australia, 2002.
13. U.Y. Nahm, M. Bilenko, and Mooney R.J. Two approaches to handling noisy variation in text mining. In *TextML'02*, pages 18–27, Sydney, 2002.
14. K. Nigam, A. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2):103–134, 2000.
15. S. Tejada, C.A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *ACM KDD'02*, pages 350–359, Edmonton, 2002.
16. W.E. Winkler. Methods for evaluating and creating data quality. *Elsevier Information Systems*, 29(7):531–550, 2004.
17. W.E. Winkler. Overview of record linkage and current research directions. Technical Report RR2006/02, US Bureau of the Census, 2006.
18. H. Yu, J. Han, and K.C.C. Chang. PEBL: positive example based learning for Web page classification using SVM. In *ACM KDD'02*, pages 239–248, Edmonton, 2002.