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 an increasingly important data preparation step in many data mining projects, as linked data can enable studies that are not feasible otherwise, or that would require expensive 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. Many modern classification techniques 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 they are then used in the second step to train a binary classifier. An experimental evaluation shows that this approach can outperform k-means clustering and 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 and government agencies, techniques that enable efficient sharing of large databases between organisations are of increasing importance in many data mining projects. Data from various sources often has to be linked in order to improve data quality and integrity, or to enrich existing data with additional information [12]. Linking entities is often challenged by the lack of entity identifiers, and thus sophisticated linkage techniques, using the available record attributes, are required [7].

For large databases, it is not feasible to compare each record from one database with all records from another database, as this process is computationally too expensive [7]. Blocking techniques are employed to reduce the number of record pair comparisons [1]. They group records into blocks, and 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 are 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 thus often a very imbalanced problem.

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 511–518, 2008.

	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 [5])

Candidate record pairs are compared using similarity functions applied to selected record attributes. These functions can be as simple as an exact string or a numerical comparison, can take typographical variations into account [2], or they can be specialised, for example, for date or time values. Each similarity function returns a numerical *matching weight*, often normalised such that 1.0 corresponds to exact similarity and 0.0 to total dissimilarity. For each compared record pair a *weight vector* is formed that contains the pair's matching weights. Using these weight vectors, record pairs are then classified into *matches*, *non-matches*, and *possible matches*, depending upon the decision model used [7].

A record pair that has equal or very similar attribute values 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 record attributes. The matching weights calculated when comparing such a pair will be 1 (or close to 1) in all weight 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 attributes.

Based on these observations, it is normally easy to accurately classify a candidate record pair as a match when its weight vector contains only matching weights close to or equal to 1, and as a non-match when its weights are all close to or equal to 0. It is however much more difficult to correctly classify a pair that has some similar and some dissimilar attribute values. In the examples shown in Fig. 1, records *R1* and *R2* are very similar to each other, and thus very likely refer to the same person. On the other hand, *R3* and *R4* are more different from each other, and it is not obvious if they refer to the same person.

It follows that it is possible, in a first step, to automatically select weight vectors that correspond to good quality training examples. 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 binary classifier for classification of all weight vectors.

This two-step approach to record pair classification has first been proposed by the author in [5], with initial experiments indicating its feasibility. The contribution of this paper is the investigation of a potential improvement to the basic approach, namely to randomly include additional weight vectors for training.

2 Related Work

In recent years, various techniques have been explored for record pair classification [7]. Among the supervised learning techniques used are decision trees [8,11] and support vector machines [10], while another approach is adaptive string similarity functions [2]. While supervised techniques normally achieve better linkage quality than unsupervised ones, their major drawback is the lack of training data (record pairs with known true match and non-match status) in many real world situations, as manual preparation of training data is time consuming and expensive. Active learning aims to overcome this problem through manual classification of only the most difficult record pairs to classify automatically [11].

Three classification approaches were compared in [8]: decision trees; k-means with three clusters (matches, possible matches and non-matches); and a hybrid approach that first clusters a sub-set of 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 k-means.

Methods similar to the proposed two-step approach have been developed for text and Web page classification [9,13], where often only a small number of positive training examples is available besides many unlabeled documents. The aim is to learn a binary classifier from positive and unlabeled examples. PEBL [13] iteratively trains a support vector machine (SVM) using the positive and negative documents furthest away from the decision boundary, while the S-EM [9] 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 in this paper.

3 Two-Step Classification

In the first step of the proposed approach, weight vectors that with high likelihood correspond to true matches and true non-matches are selected as training examples. In the second step, these training examples are used to train a binary 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 [5]. In the first approach, weight vectors that have all their vector elements within a certain distance threshold to the exact similarity or total dissimilarity values, respectively, will be selected. For example, using the weight vectors from Fig. 1 and a threshold of 0.2, only *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 next vectors closest to them.

The notation used below is a follows. It is assumed that candidate record pairs are compared using d similarity 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, with $|\cdot|$ denoting the number of elements in a set. All comparison functions are assumed to return normalised matching weights between 0 (total dissimilarity) and 1 (exact similarity), i.e. $0 \leq \mathbf{w}_i[j] \leq 1, 1 \leq j \leq d, \forall \mathbf{w}_i \in \mathbf{W}$. The weight vector containing exact similarities in all vector elements (i.e. corresponding to an exact match) is denoted by $\mathbf{m}$ (with $\mathbf{m}[j] = 1, 1 \leq j \leq d$), and the vector with only dissimilarities by $\mathbf{n}$ (with $\mathbf{n}[j] = 0, 1 \leq j \leq d$). In the training example selection step, weight vectors from $\mathbf{W}$ will be inserted into the match training example set, $\mathbf{W}_M$, and the non-match training example set, $\mathbf{W}_N$. Generally, 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 $0 < t_M, t_N < 1$), are used 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: $\mathbf{W}_M = \{\mathbf{w}_i \in \mathbf{W} : (\mathbf{m}[j] - \mathbf{w}_i[j]) \leq t_M, 1 \leq j \leq d\}$, and $\mathbf{W}_N = \{\mathbf{w}_i \in \mathbf{W} : (\mathbf{n}[j] + \mathbf{w}_i[j]) \leq t_N, 1 \leq j \leq d\}$.

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. The training sets $\mathbf{W}_M$ and $\mathbf{W}_N$ are formed according to: $\mathbf{W}_M = \{\mathbf{w}_i \in \mathbf{W}, \mathbf{w}_k \notin \mathbf{W}_M : dist(\mathbf{m}, \mathbf{w}_i) < dist(\mathbf{m}, \mathbf{w}_k)\}$, and $\mathbf{W}_N = \{\mathbf{w}_i \in \mathbf{W}, \mathbf{w}_k \notin \mathbf{W}_N : dist(\mathbf{w}_i, \mathbf{n}) < dist(\mathbf{w}_k, \mathbf{n})\}$, with *dist* being a distance function (like Euclidean distance), $x_M = |\mathbf{W}_M|$, and $x_N = |\mathbf{W}_N|$.

Given the number of true non-matches in $\mathbf{W}$ is often much larger than the number of true matches [7], more weight vectors are selected into $\mathbf{W}_N$ than into $\mathbf{W}_M$. An estimation of the ratio r of matches to non-matches is calculated using the number of records in the two data sets to be linked, $\mathbf{A}$ and $\mathbf{B}$, and the total 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 examples selected in the first step are likely linearly separable, because $\mathbf{W}_M$ and $\mathbf{W}_N$ only contain weight vectors that are either close to $\mathbf{m}$ or close to $\mathbf{n}$, and also because usually not all weight vectors from $\mathbf{W}$ are 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 [9], adding a small number of randomly selected weight vectors from this 'gap' into $\mathbf{W}_M$ and $\mathbf{W}_N$ should improve classification accuracy, as

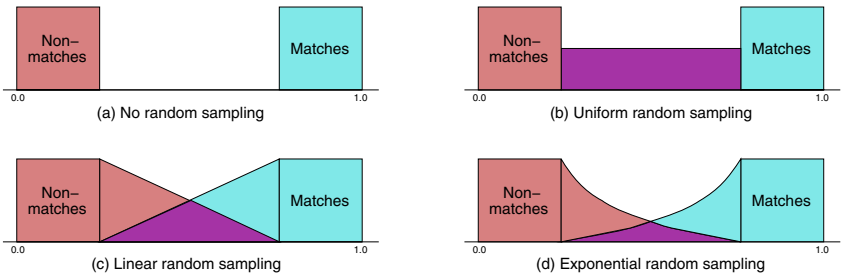

Fig. 2. Possible methods for random sampling of additional weight vectors (assumed to be 1-dimensional vectors)

the training sets will then contain a more realistic distribution of weight vectors. The random sampling of weight vectors should be done such that vectors closer to $\mathbf{m}$ are more likely included into $\mathbf{W}_M$, while vectors closer to $\mathbf{n}$ are more likely selected into $\mathbf{W}_N$. 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. These sampling methods will be evaluated experimentally below.

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 SVM classifier [3] will be evaluated, because this technique is known to be robust to noisy data.

4 Experimental Evaluation

The proposed two-step approach 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 a classification threshold that minimises both false matches and false non-matches. The second 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 [3]. The third method is k-means, with the weight vectors being clustered into matches and non-matches. Three distance measures (Manhattan, Euclidean and L_{inf}) were evaluated. All experiments were conducted using 10-fold cross validation.

All classifiers were implemented in the *Febrl* [6] open source record linkage system, which is written in Python. The *libsvm* library was used for the SVM classifier [3]. All experiments were run on a Pentium 3 GHz CPU with 2 GBytes of main memory, running Linux 2.6.20 and using Python 2.5.1.

Table 1. Data sets used in experiments. See Sect. 4 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
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

As summarised in Table 1, experiments were conducted using two real data sets from the *SecondString* toolkit[1] and four synthetic data sets containing names and addresses created with the *Febrl* data set generator [4]. The synthetic data sets are based on real-world frequency tables, and contain 60% original and 40% duplicate records (with randomly generated duplicates [5]). Standard blocking [1] was applied to reduce the number of record pair comparisons, and the Winkler string comparator [12] was used for comparing name and address values.

The quality and complexity of the compared record pairs is shown in Table 1 using the measures *pairs completeness* (number of true matched record pairs generated by blocking divided by the total number of true matched pairs) and *reduction ratio* (one minus the number of record pairs generated by blocking divided by the total number of pairs) [7,8]. The F-measure, the harmonic mean of precision and recall, is used to measure classifier performance, as accuracy is not suitable for evaluating record pair classification due to the imbalanced distribution of matches and non-matches [7]. The quality of the training example sets generated in step one, 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 $\mathbf{W}_N$|/|$\mathbf{W}_N$|).

4.1 Training Example Quality

As Table 2 shows, the quality of $\mathbf{W}_M$ and $\mathbf{W}_N$ is very good in most cases. For the threshold based approach, setting $t_M, t_N = 0.5$ achieved the best results, while a lower threshold can produce empty training sets (denoted by '–'), if all weight vectors have at least one matching weight with a similarity value above the selected threshold Nearest-based selection overcomes this problem, and generally results in very good quality training example sets [5].

4.2 Classification Performance

Figure 3 shows the F-measure results of two data sets (due to limited space not all results can be shown) over the parameter settings described in Sect. 4. The

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

Table 2. Quality of training examples generated in the first step, adapted from [5]. 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.

Data sets	Thresholds			Nearest		
	0.3	0.5	0.7	1%	5%	10%
Census	100/–	96.2/100	73.4/100	100/100	100/100	100/100
Restaurant	98.5/–	4.5/100	0.19/100	100/100	76.7/100	58.6/100
DS-Gen-A	100/100	100/100	100/99.0	100/100	100/95.9	100/95.5
DS-Gen-B	100/100	100/100	99.8/99.4	100/99.0	100/98.4	100/98.2
DS-Gen-C	100/100	100/100	98.0/99.7	100/99.7	100/99.7	100/99.6
DS-Gen-D	100/99.7	100/100	95.5/99.9	100/99.8	100/99.8	100/99.7

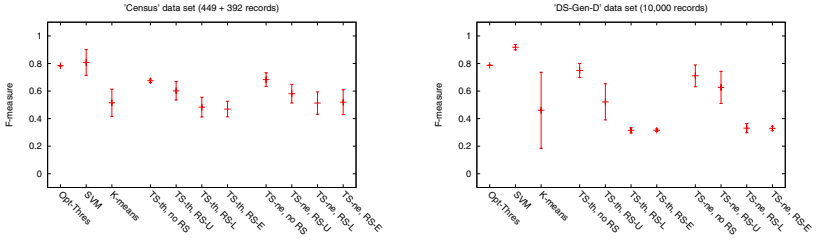

Fig. 3. F-measure results (averages and standard deviations). 'TS' stands for two-step, with 'ne' for nearest and 'th' for threshold based training example selection. 'RS' denotes the random selection: 'U' for uniform, 'L' for linear and 'E' for exponential.

four random selection methods described in Sect. 3.2 are shown for the two-step classifier (with both 1% and 10% randomly added weight vectors evaluated). As expected, both supervised classifiers (optimal threshold and SVM) performed best. The two-step classifier outperformed k-means only without random selection of additional weight vectors. Contrary to expectations, all random inclusion methods worsen the quality of the training sets and result in significantly reduced classification performance for the two data sets shown (for other data sets, slightly better classification results were achieved for linear or exponential random selection compared to no random selection). These results indicate that, unlike the random inclusion of 'spy' documents for semi-supervised text classification [9], inclusion of additional randomly selected weight vectors in the two-step approach is not improving record pair classification.

Given that normally only a portion of all weight vectors from $\mathbf{W}$ are used for training in the two-step classifier, the training time will be reduced. For example, if only 10% of all weight vectors are selected for training, then the training step should be around ten times faster compared to using all weight vectors in $\mathbf{W}$, improving the scalability of record pair classification for large data sets.

5 Conclusions and Future Work

The discussed two-step approach allows unsupervised record pair classification with often better linkage quality than k-means clustering. Contrary to expectations, the inclusion of randomly selected additional weight vectors did not increased classification performance. Future work will include the evaluation of an approach that iteratively refines the training example sets by including the strongest classified matches and non-matches in each iteration, similar to the PEBL classifier developed for text and Web page classification [13].

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.

References

1. Baxter, R., Christen, P., Churches, T.: A comparison of fast blocking methods for record linkage. In: ACM KDD 2003 workshop on Data Cleaning, Record Linkage and Object Consolidation, Washington DC, pp. 25–27 (2003)
2. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: ACM KDD 2003, Washington DC, pp. 39–48 (2003)
3. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. Manual, Department of Computer Science, National Taiwan University (2001)
4. Christen, P.: Probabilistic data generation for deduplication and data linkage. In: Gallagher, M., Hogan, J.P., Maire, F. (eds.) IDEAL 2005. LNCS, vol. 3578, pp. 109–116. Springer, Heidelberg (2005)
5. Christen, P.: A two-step classification approach to unsupervised record linkage. In: AusDM 2007, CRPIT, Gold Coast, Australia, vol. 70 (2007)
6. Christen, P.: Febrl - a freely available record linkage system with a graphical user interface. In: HDKM 2008, CRPIT, Wollongong, Australia, vol. 80 (2008)
7. Christen, P., Goiser, K.: Quality and complexity measures for data linkage and deduplication. In: Quality Measures in Data Mining. Studies in Computational Intelligence, vol. 43, pp. 127–151. Springer, Heidelberg (2007)
8. Elfeky, M.G., Verykios, V.S., Elmagarmid, A.K.: TAILOR: A record linkage toolbox. In: ICDE 2002, San Jose, pp. 17–28 (2002)
9. Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: ICML 2002, Sydney, Australia, pp. 387–394 (2002)
10. Nahm, U.Y., Bilenko, M., Mooney, R.J.: Two approaches to handling noisy variation in text mining. In: TextML 2002, Sydney, pp. 18–27 (2002)
11. Tejada, S., Knoblock, C.A., Minton, S.: Learning domain-independent string transformation weights for high accuracy object identification. In: ACM KDD 2002, Edmonton, pp. 350–359 (2002)
12. Winkler, W.E.: Methods for evaluating and creating data quality. Elsevier Information Systems 29(7), 531–550 (2004)
13. Yu, H., Han, J., Chang, K.C.C.: PEBL: positive example based learning for Web page classification using SVM. In: ACM KDD 2002, Edmonton (2002)