Efficient Interactive Training Selection for Large-Scale Entity Resolution

Qing Wang(✉), Dinusha Vatsalan, and Peter Christen

Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia
{qing.wang,dinusha.vatsalan,peter.christen}@anu.edu.au

Abstract. Entity resolution (ER) has wide-spread applications in many areas, including e-commerce, health-care, the social sciences, and crime and fraud detection. A crucial step in ER is the accurate classification of pairs of records into matches (assumed to refer to the same entity) and non-matches (assumed to refer to different entities). In most practical ER applications it is difficult and costly to obtain training data of high quality and enough size, which impedes the learning of an ER classifier. We tackle this problem using an interactive learning algorithm that exploits the cluster structure in similarity vectors calculated from compared record pairs. We select informative training examples to assess the purity of clusters, and recursively split clusters until clusters pure enough for training are found. We consider two aspects of active learning that are significant in practical applications: a limited budget for the number of manual classifications that can be done, and a noisy oracle where manual labeling might be incorrect. Experiments using several real data sets show that manual labeling efforts can be significantly reduced for training an ER classifier without compromising matching quality.

Keywords: Data matching · Record linkage · Deduplication · Active learning · Noisy oracle · Hierarchical clustering · Interactive labeling

1 Introduction

Entity resolution (ER), also known as data matching, record linkage, or duplicate detection, is the process of identifying and matching records that correspond to the same entities from one or more databases [6]. As the databases to be matched generally do not include entity identifiers, ER has to be based on the available attributes, for example, personal names, addresses and dates of birth. In the past decades, ER has attracted much interest from various application domains, the most prominent being health, census statistics, e-commerce, national security and digital libraries. For recent surveys see [6,14].

The core steps in ER in their most basic form consist of the pair-wise comparison of records using functions that calculate numerical similarities between

This research was partially funded by the Australian Research Council (ARC), Veda, and Funnelback Pty. Ltd., under Linkage Project LP100200079.

© Springer International Publishing Switzerland 2015
T. Cao et al. (Eds.): PAKDD 2015, Part II, LNAI 9078, pp. 562–573, 2015.
DOI: 10.1007/978-3-319-18032-8_44

attribute values, followed by either an unsupervised or supervised classification of pairs of records into matches and non-matches [6,14]. The comparison of attribute values used in ER is commonly based on approximate string comparison functions that return a normalized similarity between 0 (totally different values) and 1 (exact matching values). For each compared record pair, a *weight vector* is calculated with the similarities over the different attributes of that pair [6].

Various supervised and unsupervised learning techniques [3,5,7,9,13] have been proposed for ER in past years. While supervised techniques generally result in much better matching quality, these techniques require training data in the form of labeled examples of true matching pairs of records that refer to the same entity, and true non-matching pairs of records that refer to different entities. While in certain, mostly academic, situations such training data are available, in most practical applications of ER actual truth data are difficult to obtain. In many cases training data have to be manually generated, a task that is known to be difficult both in terms of cost and quality [6]. The traditional way of selecting training data is to use random sampling. However, from a robust statistical point of view, random sampling needs to select a significantly large number of examples for guaranteeing the quality of training data, which was also verified in our experiments discussed in Sect. 5. Another difficulty of using random sampling is caused by the imbalance of the ER problem, as the vast majority of record pairs will correspond to non-matches [6]. Two challenges thus stand out in particular when training data are to be manually generated over large-scale data sets: (1) How can we ensure "good" examples are selected for training? (2) How can we minimize the user's burden of labeling examples?

Active learning is a promising approach for selecting training data [1,19,22]. The central idea is to reduce the labeling efforts through *actively* choosing *informative* or *representative* examples [16]. In doing so, instead of choosing a large quantity of examples to label as is required for fully supervised learning, active learning only selects examples based on the hints from previously labeled examples, which can often yield a training set that is small but still sufficient for supporting accurate classification.

Although successful, existing active learning methods for ER have limitations in achieving efficient training for large-scale data sets. Most of these methods are grounded on a monotonicity assumption – a record pair with higher similarity is more likely to represent the same entity than a pair with lower similarity. This assumption is valid in some real-world applications but does not generally hold, as we will illustrate in Sect. 3. Thus, two difficult issues arise in selecting training data: (1) How do we know whether the monotonicity assumption holds on a data set since training data are not available? (2) How can we effectively select training data when the monotonicity assumption does not hold?

In this paper we develop a generic active learning method for efficiently selecting ER training data over large data sets. Unlike other works, we do not rely on the monotonicity assumption. Instead, our method exploits the cluster structure in data through active learning, which can circumvent the first issue above, meanwhile solving the second issue. The basic idea of our method is illustrated in

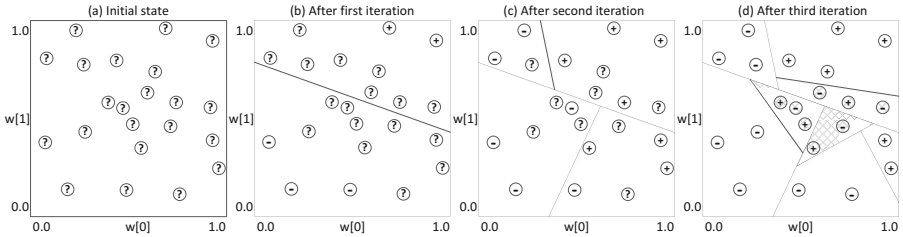

Fig. 1. Example of the training selection process with 2-dimensional weight vectors. Weight vectors that have been labeled as matches are shown with a $\oplus$, non-matches with a $\ominus$, and unlabeled ones with a circled question mark. The shaded areas in (d) represent fuzzy clusters, while the other areas in (d) represent pure clusters, which are to be used as training data for an ER classifier.

Fig. 1, where (a) shows weight vectors that are generated from pair-wise record comparisons, and the labels of these weight vectors are unknown. Then, (b) to (d) show how the weight vectors are interactively selected and manually classified, and how the set of weight vectors is recursively split into smaller clusters until each cluster is classified as being *pure* or *fuzzy* (to be formally defined in Sect. 4) based on the label *purity* of its informative weight vectors. During this process, the training set is interactively constructed by gathering the weight vectors from pure clusters.

We make the following contributions in this paper. (1) We develop an interactive training method which can be applied to ER training tasks without prior knowledge of the match and non-match distributions of the underlying data sets. (2) Our training method incorporates a budget-limited noisy human oracle, which ensures: (i) the overall labeling efforts can be controlled at an acceptable level and as specified by the user, and (ii) the accuracy of labeling provided by human experts can be simulated. This is in contrast to existing active learning methods for ER which often assume a perfect and unlimited labeling process [20]. (3) We experimentally evaluate our method on four real-world data sets from different application domains.

In the following section we discuss related work. In Sect. 3 we present the problem and building blocks of our approach, which we describe in detail in Sect. 4. We experimentally evaluate our approach in Sect. 5, and conclude the paper in Sect. 6 with an outlook to future work.

2 Related Work

Active learning has previously been studied in many problem domains, such as text classification and speech recognition [20]. In the area of ER, active learning has been explored for learning ER classifiers, which classify pairs of records as matches or non-matches through actively selecting a reduced number of examples

for labeling [1,2,10,19,22]. In the following we provide a brief overview of work that relates to our study in this paper.

Early work on active learning strategies for finding informative or representative examples typically used disagreement between multiple classifiers. For example, a committee of classifiers was used to identify the most representative examples for labeling [19,22], i.e., labeling is iteratively required for pairs of records where the classifiers return contradictory labels for the same example. Sampling based active learning and its bias have been discussed in [11].

Later work has concentrated on the learning quality guarantee, that typically has a linear combination of the two measures precision and recall as the learning objective. For example, in [1,2], given a minimum precision specified by the user, a learned classifier aims to have a precision greater than the minimum precision and a recall close to the best possible. Compared with these active learning techniques, our algorithm has several interesting properties: (1) providing an integrated view on labeling budget control and quality guarantee, (2) using interactive purity-based classification to reduce examples for labeling, and (3) not relying on the monotonicity assumption for improving quality.

To improve the efficiency of active learning, two techniques have commonly been used. One is to incorporate blocking or indexing [6] into the learning process with parameters that are tuned manually [1,2] or semi-automatically [10]. Blocking in ER is the process of dividing data sets into smaller blocks according to some criteria so that only records within the same block are compared with each other. In principle, existing blocking techniques can be easily incorporated into active learning algorithms as a pre-processing step before learning. The second technique is to optimize active learning algorithms under certain distribution assumptions, such as the monotonicity [1] and low noise [2] assumptions. Our training method is completely independent of any assumption concerning the data set or any blocking technique used, which makes our method more generally applicable.

A number of studies have attempted to control labeling noise using certain strategies. Repeated labeling strategies were investigated in [21], including round-robin repeated labeling and selective repeated labeling based on the uncertainty of labels. In [12], a combined strategy was proposed, which selects examples that are more likely to be correctly labeled yet still provide high quality information, and examples that are most likely to have been incorrectly labeled. In [23] the most reliable oracle was selected among multiple noisy oracles for labeling. In this paper, we explore active learning in the presence of a noisy human oracle, which allows us to simulate the challenging manual clerical labeling process in real-world ER applications.

3 Problem Statement and Building Blocks

We study the problem of reducing the labeling costs for selecting training data, while keeping the quality of ER classification at a high level. In contrast to the works of [1,2], we do not rely on the monotonicity assumption since it does

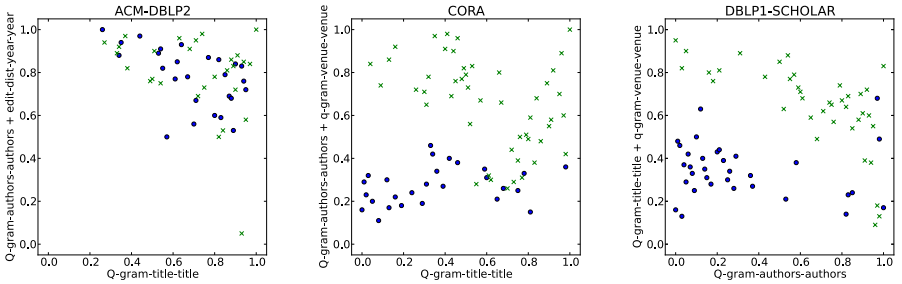

Fig. 2. Examples where the monotonicity assumption of similarities does not hold: non-matches with the highest similarity (denoted by light green crosses) and matches with the lowest similarity (denoted by dark blue dots)

not generally hold for ER. This is evident from the plots in Fig. 2, which show the non-matches with the highest similarity and matches with the lowest similarity from three of the real-world data sets we used in our experimental evaluation in Sect. 5. To address this problem, we propose an active learning approach that, given a set of weight vectors and a classifier, recursively splits the weight vectors into clusters, and classifies these as being matches or non-matches if the purity of informative weight vectors in a cluster is higher than a specified threshold. In the following we present the building blocks of our proposed approach.

Let $\mathbf{R}$ be a set of records from one or more data sets, each $r \in \mathbf{R}$ having a set of attributes. We use $r.A$ to refer to the value of an attribute A in a record r. Given two records $r_1, r_2 \in \mathbf{R}$ and an attribute A of r_1 and r_2, a *similarity weight* of A between r_1 and r_2 is a value in $[0, 1]$, denoted as $f(r_1.A, r_2.A)$, where f is a similarity function [6] that quantifies the similarity between $r_1.A$ and $r_2.A$. Taking the edit distance similarity function f_{ed} for example [6], $f_{ed}(r_1.fname, r_2.fname) = 1.0 - \frac{3}{6} = 0.5$, where $r_1.fname = Rob$ and $r_2.fname = Robert$.

For a set $\mathbf{A} = \{A_1, \ldots, A_n\}$ of attributes selected for performing ER tasks, each compared pair (r_1, r_2) of records that has the attributes $\mathbf{A}$ results in a *weight vector* $\langle a_1, \ldots, a_n \rangle \in [0, 1]^n$ over $\mathbf{A}$, where a_i is the similarity weight of A_i between r_1 and r_2 $(i = 1, \ldots, n)$. For example, the pair (r_1, r_2) of records over the attributes $\{fname, sname, age\}$ with $r_1.fname = Rob$, $r_1.sname = Smith$, $r_1.age = 30$, $r_2.fname = Robert$, $r_2.sname = Smith$ and $r_2.age = 31$ may correspond to a weight vector $\langle 0.5, 1, 0.5 \rangle$. A *weight vector set* $\mathbf{W}$ over $\mathbf{A}$ consists of all the weight vectors over $\mathbf{A}$ to which the pairs of records in $\mathbf{R}$ correspond. A *cluster* $\mathbf{W}_i \subseteq \mathbf{W}$ is a subset of weight vectors in $\mathbf{W}$. A *partition* of $\mathbf{W}$ is a set $\{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$ of pairwise disjoint clusters whose union contains all the weight vectors in $\mathbf{W}$, i.e. $\mathbf{W}_i \cap \mathbf{W}_j = \emptyset$ for $1 \leq i \neq j \leq m$, and $\bigcup_{1 \leq i \leq m} \mathbf{W}_i = \mathbf{W}$.

We consider a noisy human oracle that simulates a non-perfect manual clerical labeling process. The main reason behind such noisy human oracles is due to the fact that human experts often have different levels of expertise for labeling

matches and non-matches [6]. Thus, depending on which human expert is asked for labeling an example, the labeling accuracy varies. A human oracle takes a set of record pairs and their corresponding weight vectors as input, and based on manual inspection of the attribute values of these records assigns each weight vector with a label. Let $\mathbf{W}_i$ be a weight vector set. Then a *human oracle* over $\mathbf{W}_i$ is a function $\zeta : \mathbf{W}_i \mapsto \{M, N\}$, where M and N are the two labels indicating *match* and *non-match* of a weight vector, respectively. Moreover, each human oracle ζ is associated with a pair $\langle bud(\zeta), acc(\zeta)\rangle$, where $bud(\zeta) > 0$ is a budget limit (b_{tot}) indicating the maximal number of weight vectors that can be labeled by ζ, and $acc(\zeta) \in [0, 1]$ is indicating the accuracy of labels provided by ζ. If $acc(\zeta) = 1$ then the oracle is *perfect*.

We view an *ER classifier* as a black-box that classifies record pairs into matches and non-matches through their corresponding weight vectors [6]. More specifically, an *ER classifier* takes as input a weight vector set $\mathbf{W}_i$ and a subset of labeled (with M and N) weight vectors $\mathbf{W}_i^T \subseteq \mathbf{W}_i$ as the training set, and generates a partition of $\mathbf{W}_i$ into $\mathbf{W}_i^M$ of matches and $\mathbf{W}_i^N$ of non-matches, with $\mathbf{W}_i^M \cap \mathbf{W}_i^N = \emptyset$. A variety of classifiers have previously been used for ER, such as decision trees [22], SVMs [7,19] and k-nearest neighbor [4], any of which can be used in our approach.

4 Recursive Interactive Training Algorithm

In this section we discuss the details of our approach. A high-level description of our interactive training approach is provided in Algorithm 1. Let $\mathbf{W}$ be a weight vector set, and $\mathbf{T}^M$ and $\mathbf{T}^N$ be the subsets of $\mathbf{W}$ that are selected into the match and non-match training sets, respectively, with $\mathbf{T}^M \cap \mathbf{T}^N = \emptyset$.

After initialization, the algorithm starts with $\mathbf{W}$ being inserted into an empty queue $\mathbf{Q}$ of clusters to be processed (line 2). The main iteration (line 4) loops as long as the queue is not empty (i.e. there are clusters to process) and the total oracle budget b_{tot} has not been fully used ($b \leq b_{tot}$). In each iteration, the first cluster $\mathbf{W}_i$ in the queue is being processed. In the first loop (with $b = 0$ indicating no manual labeling has been done), the INIT_SELECT() function is used to select a first set of weight vectors $\mathbf{S}_i \subseteq \mathbf{W}_i$ to be manually classified by the oracle, while in sub-sequent iterations the MAIN_SELECT() function is used. Different approaches for these selection functions will be described in Sect. 4.1. In general, a selection function selects k informative weight vectors $\mathbf{S}_i$ from a cluster $\mathbf{W}_i$ (lines 7 or 9).

The weight vectors in $\mathbf{S}_i$ are then manually classified by the human oracle (line 11) into a match set $\mathbf{T}_i^M$ and non-match set $\mathbf{T}_i^N$, which are added to the final training sets $\mathbf{T}^M$ and $\mathbf{T}^N$, respectively (line 12). The used budget is also increased (line 10) by the number of manually classified weight vectors $|\mathbf{S}_i|$. Then, the purity p_i of the cluster is calculated (line 13), as will be described further below. All weight vectors in the cluster are added into one of the training sets (lines 14 to 17) if the cluster is pure enough ($p_i \geq p_{min}$); otherwise, the cluster requires further splitting if it is larger than a minimum cluster size c_{min}, and

Algorithm 1. *Recursive interactive training algorithm*

Input:
- A weight vector set: $\mathbf{W}$
- Budget limit: b_{tot}
- Minimum purity threshold: p_{min}
- Initial selection function: INIT_SELECT$(\cdot)$
- Main selection function: MAIN_SELECT$(\cdot)$
- Human oracle for labeling: ORACLE$(\cdot)$
- Number of weight vectors to select for labeling: k
- Minimum size of a cluster: c_{min}
- Classifier function used for splitting clusters: CLASSIFIER

Output:
- Match and non-match training set $\mathbf{T}^M$ and $\mathbf{T}^N$

1: $\mathbf{T}^M = \emptyset, \mathbf{T}^N = \emptyset$ // Initialize training sets as empty
2: $\mathbf{Q} = [\mathbf{W}]$ // Initialize queue of clusters
3: $b = 0$ // Initialize number of manually labeled examples
4: **while** $\mathbf{Q} \neq \emptyset$ **and** $b \leq b_{tot}$ **do**:
5: $\mathbf{W}_i = \mathbf{Q}.pop()$ // Get first cluster from queue
6: **if** $b = 0$ **then**:
7: $\mathbf{S}_i$ = INIT_SELECT$(\mathbf{W}_i, k)$ // Initial selection of weight vectors
8: **else**:
9: $\mathbf{S}_i$ = MAIN_SELECT$(\mathbf{W}_i, k)$ // Select informative weight vectors
10: $b = b + |\mathbf{S}_i|$ // Update number of manual labeling done so far
11: $\mathbf{T}_i^M, \mathbf{T}_i^N, p_i$ = ORACLE$(\mathbf{S}_i)$ // Manually classify selected weight vectors
12: $\mathbf{T}^M = \mathbf{T}^M \cup \mathbf{T}_i^M$; $\mathbf{T}^N = \mathbf{T}^N \cup \mathbf{T}_i^N$; $\mathbf{W}_i = \mathbf{W}_i \setminus (\mathbf{T}_i^M \cup \mathbf{T}_i^N)$
13: **if** $p_i \geq p_{min}$ **then**:
14: **if** $|\mathbf{T}_i^M| > |\mathbf{T}_i^N|$ **then**:
15: $\mathbf{T}^M = \mathbf{T}^M \cup \mathbf{W}_i$ // Add whole cluster to match training set
16: **else**:
17: $\mathbf{T}^N = \mathbf{T}^N \cup \mathbf{W}_i$ // Add whole to non-match training set
18: **else if** $|\mathbf{W}_i| > c_{min}$ **and** $b \leq b_{tot}$ **then**: // Low purity, split cluster further
19: **if** $\mathbf{T}_i^M \neq \emptyset$ **and** $\mathbf{T}_i^N \neq \emptyset$ **then**:
20: CLASSIFIER.$train(\mathbf{T}_i^M, \mathbf{T}_i^N)$ // Train classifier
21: $\mathbf{W}_i^M, \mathbf{W}_i^N$ = CLASSIFIER.$classify(\mathbf{W}_i)$ // Classify current cluster
22: $\mathbf{Q}.append(\mathbf{W}_i^M)$; $\mathbf{Q}.append(\mathbf{W}_i^N)$ // Append new clusters to queue
23: **return** $\mathbf{T}^M$ and $\mathbf{T}^N$

the total oracle budget b_{tot} has not been fully used, and if $\mathbf{T}_i^M$ and $\mathbf{T}_i^N$ are not empty (lines 18 to 22). If $\mathbf{T}_i^M$ and $\mathbf{T}_i^N$ are both not empty, they will be used to train a classifier for the current cluster $\mathbf{W}_i$ (line 20). The splitting of $\mathbf{W}_i$ (line 21) leads to two smaller clusters $\mathbf{W}_i^M$ and $\mathbf{W}_i^N$ of matches and non-matches, respectively, which are then added to the queue (line 22). In principle, the two smaller clusters $\mathbf{W}_i^M$ and $\mathbf{W}_i^N$ should have a higher purity compared to $\mathbf{W}_i$. Clusters that are small ($|\mathbf{W}_i| \leq c_{min}$) and not pure are not considered for inclusion into the final training sets.

The algorithm thus generates a partition of $\mathbf{W}$ such that the weight vectors in each pure cluster are selected into the training sets, i.e., the weight vectors from a match cluster into $\mathbf{T}^M$ and the ones from a non-match cluster into $\mathbf{T}^N$, while the weight vectors in fuzzy clusters (those too small for further splitting and not pure enough) are discarded.

The *purity* p_i of a cluster $\mathbf{W}_i$ is calculated based on the classification done by the human oracle (line 11) using the manually classified weight vector set $\mathbf{S}_i$ as the proportion of classified weight vectors that have the majority label:

$$p_i = purity(\mathbf{W}_i) = max\left(\frac{|\mathbf{T}_i^M|}{|\mathbf{T}_i^M \cup \mathbf{T}_i^N|}, \frac{|\mathbf{T}_i^N|}{|\mathbf{T}_i^M \cup \mathbf{T}_i^N|}\right), \tag{1}$$

where $|\mathbf{T}_i^M \cup \mathbf{T}_i^N| = |\mathbf{S}_i|$. For a given purity threshold $p_{min} \in [0.5, 1]$, a cluster $\mathbf{W}_i$ is labeled as *pure* if $purity(\mathbf{W}_i) > p_{min}$; otherwise $\mathbf{W}_i$ is labeled as *fuzzy*.

4.1 Weight Vector Selection Methods

The informativeness of selected weight vectors crucially influences the quality of the final generated training sets $\mathbf{T}^M$ and $\mathbf{T}^N$. Therefore, the selection methods in Algorithm 1 need to be carefully chosen. Here we propose three methods for the INIT_SELECT function and four methods for the MAIN_SELECT functions.

Let $\mathbf{W}_i$ be a set of weight vectors over the predefined set $\mathbf{A}$ of attributes. For the initial selection (line 7 in Algorithm 1) using INIT_SELECT we consider: (1) *Far:* Farthest-first weight vectors with random initialization [15], based on the farthest first clustering algorithm which selects the k weight vectors from $\mathbf{W}_i$ that are farthest apart from each other. The idea of this approach is to start with a selection of weight vectors with the highest possible variety. (2) *01:* Weight vectors that are closest to the two corners $[1]^{|\mathbf{A}|}$ and $[0]^{|\mathbf{A}|}$. These are most likely to represent matches and non-matches, respectively, as they correspond to weight vectors closest to exact matching and totally different record pairs [7]. (3) *Corner:* Weight vectors that are closest to all corners $\{[a_1, \ldots, a_{|\mathbf{A}|}]|a_i \in \{0, 1\}$ for $i = 1, \ldots, |\mathbf{A}|\}$, where there are $2^{|\mathbf{A}|}$ corners in total. This approach combines the ideas of both **Far** and **01**, selecting weight vectors with the highest possible variety in terms of all the attributes in $\mathbf{A}$.

Analogously, for follow-up selection of weight vectors from a cluster $\mathbf{W}_i$ during the main iteration of Algorithm 1 (line 9) using MAIN_SELECT we consider: (1) *Ran:* A random selection of k weight vectors. We use this as a baseline in our experiments to evaluate the effectiveness of the other selection methods. (2) *Far:* Farthest-first weight vectors selection within a cluster, as done in the *Far* initial selection method. This will give us weight vectors at the outer boundary of a cluster. (3) *Far-Med:* Here we select the $k-1$ farthest apart weight vectors from $\mathbf{W}_i$, and additionally we add the medoid weight vector closest to the center of the cluster. The idea is to not just manually classify pairs at the boundary of a cluster, but also one weight vector in its center to get a better picture of the distribution of matches and non-matches in the cluster.

In the following section we evaluate these different methods in combination with different parameter settings on several real-world data sets.

5 Experimental Evaluation

We conducted experiments on four data sets: ACM-DBLP [17], CORA[1], DBLP-Google Scholar (DBLP-GS) [17], and the North Carolina Voter Registration (NCVR) database[2]. The characteristics of these data sets are summarized in Table 1. As can be seen, all data sets exhibit a high to very high class imbalance

[1] Available from: http://secondstring.sourceforge.net
[2] Available from: ftp://alt.ncsbe.gov/data/

Table 1. Characteristics of data sets used in experiments

Data set name(s)	Number of records	Number of unique weight vectors	Class imbalance	Time for pair-wise comparisons
NCVR	224,073 / 224,061	3,495,580	1 : 27	441.6 sec
CORA	1,295	286,141	1 : 16	47.0 sec
DBLP-GS	2,616 / 64,263	8,124,258	1 : 3273	963.1 sec
ACM-DBLP	2,616 / 2,294	687,910	1 : 1785	95.3 sec

between true matches and true non-matches. We used the *Febrl* open source record linkage system for the pair-wise linkage step, together with a variety of blocking/indexing and string comparison functions [8]. The output of this step are sets of weight vectors of the compared record pairs, and their known true labels (match or non-match).

The following parameter variations were used in our experiments: minimum purity threshold $p_{min} = [0.95, 0.9, 0.85, 0.8, 0.75]$, oracle accuracy $acc(\zeta) = [1.0, 0.95, 0.9, 0.85, 0.8, 0.75]$, total budget $b_{tot} = [100, 200, 500, 1,000, 2,000, 5,000, 10,000]$, number of weight vectors selected $k = [9, 19, 49, 69, 99]$, and the different initial selection (*Far*, *01* and *Corner*) and selection (*Ran*, *Far* and *Far-Med*) methods discussed in the previous section. The classifiers used for splitting weight vectors were decision trees (DTree) with entropy and information gain [18]. Default values for the parameters were set to minimum purity $p_{min} = 0.95$, oracle accuracy $acc(\zeta) = 1.0$, number of weight vectors selected $k = 49$ for the CORA data set and $k = 69$ for the other data sets, total budget $b_{tot} = 1,000$ for the CORA data set and $b_{tot} = 5,000$ for the other data sets, minimum cluster size $c_{min} = 50$, initial selection method *01* and selection method *Far*, as these settings resulted in the best quality based on a set of pre-experiments.

We evaluated the effectiveness of our approach using the F-measure [6], and the efficiency using the time required for the classification. The baseline approaches we used to compare with our approach (which we refer as DTree-AL) were: (1) fully supervised decision tree (DTree-S), (2) fully supervised support vector machines with linear and polynomial kernels (SVM-S), (3) unsupervised automatic k-nearest neighbor clustering (kNN-US) [7], (4) unsupervised k-means clustering (kMeans-US), and (5) unsupervised farthest first clustering (Far-US) [8]. Our proposed active learning approach and the baseline approaches are implemented in Python 2.7.3, and we ran all experiments on a server with 6-core 64-bit Intel Xeon 2.4 GHz CPUs, 128 GBytes of memory and running Ubuntu 14.04. The programs and test data sets are available from the authors.

We first evaluated how different values for the six main parameters of our approach (i.e., p_{min}, $acc(\zeta)$, b_{tot}, k, INIT_SELECT methods, and MAIN_SELECT methods) affect the quality of the classification results. Fig. 3 (a) shows the F-measure of our approach for different minimum purity thresholds (p_{min}). F-measure increases with an increasing p_{min} since a higher purity of cluster requirement results in more accurately classified clusters. As expected the F-measure also increases when the accuracy of the oracle ($acc(\zeta)$) increases (Fig. 3 (b)).

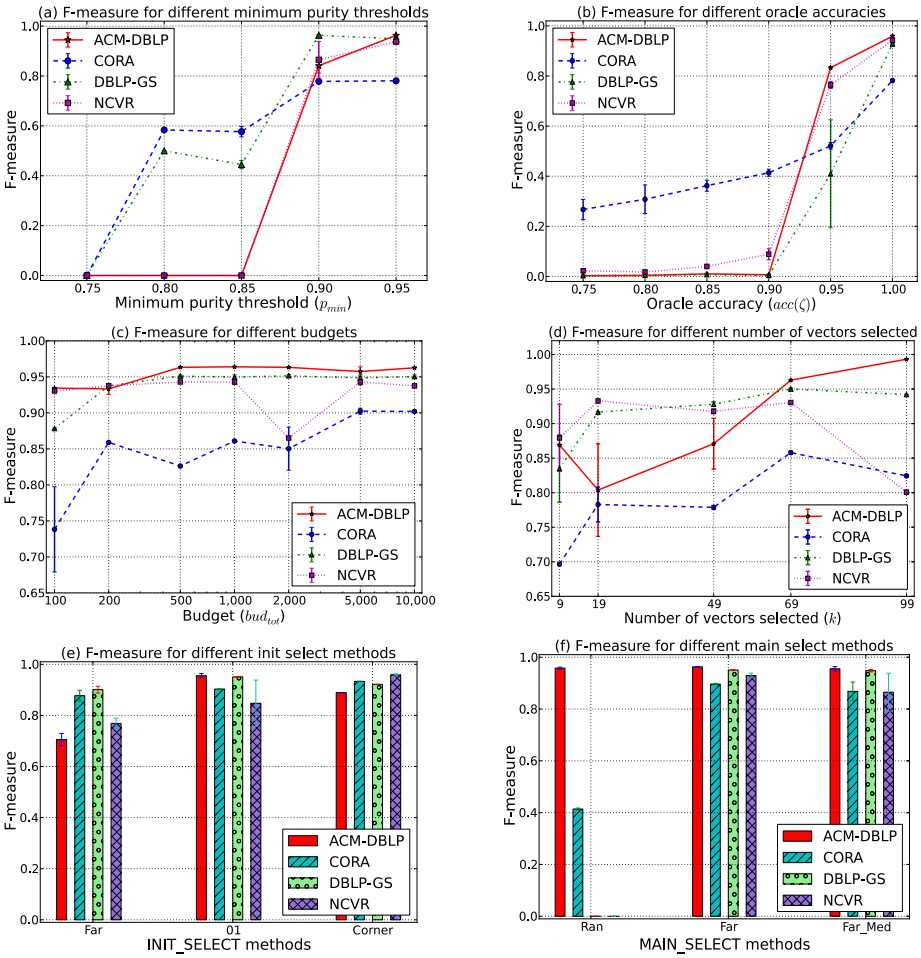

Fig. 3. F-measure against (a) minimum purity threshold, (b) oracle accuracy, (c) budget, (d) number of weight vectors selected, (e) different initial selection methods and (f) different main selection methods averaged over the results of all classifiers

F-measure increases with larger budgets (b_{tot}) and larger number of weight vectors selected (k) as can be seen from Fig. 3 (c) and (d), respectively. Larger budgets allow more vectors to be manually labeled, and a larger number of weight vectors selected from each cluster can represent the clusters more effectively, resulting in increased F-measure. However, as can be seen when $k = 99$, a smaller number of clusters can be manually assessed with larger k, potentially leading to lower F-measure. An interesting result is that a high F-measure (of ≥ 0.8) is achieved on all data sets even with a small budget size of $b_{tot} = 200$.

Among the three initial selection methods, *01* comparatively performs well, though all three methods achieve high F-measure on all four data sets except the

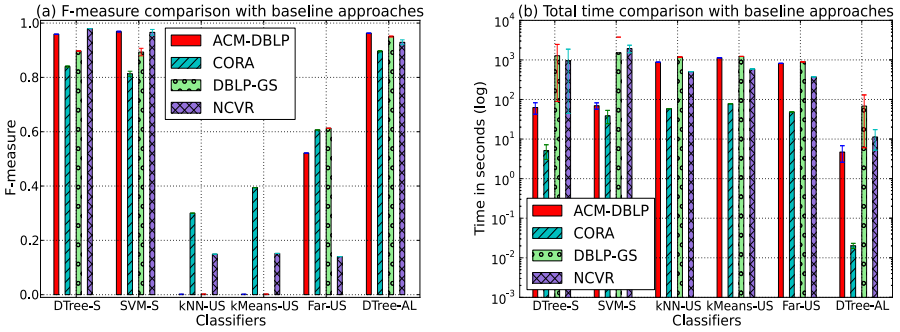

Fig. 4. Comparison of (a) F-measure and (b) total required time (log scale) of our active learning (AL) approach with different baseline supervised (S) and unsupervised (US) classifiers, averaged over the results of all variations of each classifier

Far method on the ACM-DBLP data set, as shown in Fig. 3 (e). The selection methods *Far* and *Far-Med* perform equally well on all four data sets, while *Ran* does not consistently perform well, particularly over two relatively large data sets DBLP-GS and NCVR, due to its random selection. (see Fig. 3 (f)).

Finally, we compared our approach with five baseline approaches as described above. Fig. 4 (a) shows the F-measure (effectiveness) of all six approaches and Fig. 4 (b) shows their total time required for classification (efficiency). The results illustrate that our active learning approach achieves significantly higher F-measure results compared to unsupervised approaches, and comparable results to fully supervised approaches, while requiring significantly lower runtime than all other approaches on all four data sets.

6 Conclusions and Future Work

We have developed an active learning approach for reducing the labeling costs in ER while achieving high linkage quality results. Experiments conducted on four real data sets validate the efficiency and effectiveness of our approach compared to both existing fully supervised and unsupervised ER classifiers.

As future work we plan to study the following two issues. First, how does the ordering of clusters (line 5 in Algorithm 1) in the queue affect the training quality? Since only a limited labeling budget is available, the number of weight vectors a human oracle can manually label is restricted. Once the labeling budget is run out, the training selection process terminates. Thus, the cluster selected for manual labeling at each iteration should be the one that can provide an optimal improvement in the quality, coverage and representativeness of the training data set. Second, how can our approach be improved if the accuracy of a human oracle is known? Knowing this accuracy may significantly affect the purity calculation of clusters. It is thus plausible to enhance the performance of our approach by taking the accuracy of a human oracle into account.

References

1. Arasu, A., Götz, M., Kaushik, R.: On active learning of record matching packages. In: ACM SIGMOD, Indianapolis, pp. 783–794 (2010)
2. Bellare, K., Iyengar, S., Parameswaran, A.G., Rastogi, V.: Active sampling for entity matching. In: ACM SIGKDD, Beijing, pp. 1131–1139 (2012)
3. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: ACM SIGKDD, Washington DC, pp. 39–48 (2003)
4. Chaudhuri, S., Ganti, V., Motwani, R.: Robust identification of fuzzy duplicates. In: IEEE ICDE, Tokyo, pp. 865–876 (2005)
5. Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Exploiting context analysis for combining multiple entity resolution systems. In: ACM SIGMOD, Providence, pp. 207–218 (2009)
6. Christen, P.: Data Matching. Data-Centric Systems and Applications. Springer (2012)
7. Christen, P.: Automatic training example selection for scalable unsupervised record linkage. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 511–518. Springer, Heidelberg (2008)
8. Christen, P.: Development and user experiences of an open source data cleaning, deduplication and record linkage system. SIGKDD Explorations **11**(1) (2009)
9. Cochinwala, M., Kurien, V., Lalk, G., Shasha, D.: Efficient data reconciliation. Information Sciences **137**(1), 1–15 (2001)
10. Dal Bianco, G., Galante, R., Heuser, C.A., Gonçalves, M.A.: Tuning large scale deduplication with reduced effort. In: SSDBM, Baltimore, p. 18 (2013)
11. Dasgupta, S., Hsu, D.: Hierarchical sampling for active learning. In: IEEE ICML, Helsinki, pp. 208–215 (2008)
12. Du, J., Ling, C.X.: Active learning with human-like noisy oracle. In: IEEE ICDM, Sydney, pp. 797–802 (2010)
13. Elfeky, M.G., Verykios, V.S., Elmagarmid, A.K.: TAILOR: a record linkage toolbox. In: IEEE ICDE, San Jose, pp. 17–28 (2002)
14. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: A survey. IEEE TKDE **19**(1), 1–16 (2007)
15. Hochbaum, D.S., Shmoys, D.B.: A best possible heuristic for the k-center problem. Mathematics of Operations Research **10**(2), 180–184 (1985)
16. Huang, S.J., Jin, R., Zhou, Z.H.: Active learning by querying informative and representative examples. In: NIPS, Vancouver, pp. 892–900 (2010)
17. Köpcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems. VLDB Endowment **3**(1–2), 484–493 (2010)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research **12**, 2825–2830 (2011)
19. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: ACM SIGKDD, Edmonton, pp. 269–278 (2002)
20. Settles, B.: Active learning literature survey, vol. 52, pp. 55–66. University of Wisconsin, Madison (2010)
21. Sheng, V.S., Provost, F., Ipeirotis, P.G.: Get another label? improving data quality and data mining using multiple, noisy labelers. In: ACM SIGKDD, Las Vegas, pp. 614–622 (2008)
22. Tejada, S., Knoblock, C.A., Minton, S.: Learning domain-independent string transformation weights for high accuracy object identification. In: ACM SIGKDD, Edmonton, pp. 350–359 (2002)
23. Wu, W., Liu, Y., Guo, M., Wang, C., Liu, X.: A probabilistic model of active learning with multiple noisy oracles. Neurocomputing **118**, 253–262 (2013)