Efficient Entity Resolution with Adaptive and Interactive Training Data Selection

Peter Christen, Dinusha Vatsalan, and Qing Wang
Research School of Computer Science, The Australian National University, Acton ACT 2601, Australia
Emails: {peter.christen, dinusha.vatsalan, qing.wang}@anu.edu.au

***Abstract*—Entity resolution (ER) is the task of deciding which records in one or more databases refer to the same real-world entities. A crucial step in ER is the accurate classification of pairs of records into matches and non-matches. In most practical ER applications, obtaining training data is costly and time consuming. Various techniques have been proposed for ER to interactively generate training data and learn an accurate classifier. We propose an approach for training data selection for ER that exploits the cluster structure of the weight vectors (similarities) calculated from compared record pairs. Our approach adaptively selects an optimal number of informative training examples for manual labeling based on a user defined sampling error margin, and recursively splits the set of weight vectors to find pure enough subsets for training. We consider two aspects of ER that are highly significant in practice: a limited budget for the number of manual labeling that can be done, and a noisy oracle where manual labels might be incorrect. Experiments on four real public data sets show that our approach can significantly reduce manual labeling efforts for training an ER classifier while achieving matching quality comparative to fully supervised classifiers.**

I. Introduction

Entity resolution (ER) is an important step in various application domains, including e-commerce, healthcare, national censuses, the social sciences, crime and fraud detection, and national security. In these domains, the databases to be matched often do not contain entity identifiers. ER therefore has to rely on available attributes, such as names and addresses for people, or titles and author names for publications.

The basic steps in ER [5] consist of the pair-wise comparison of records, using functions that calculate similarities between attribute values, followed by the classification of the compared record pairs into matches (assumed to refer to the same entity) and non-matches (assumed to refer to different entities). The comparison of attribute values is commonly based on approximate string matching 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 formed that contains the similarities over the different attributes of that pair [5].

Various classification techniques have been proposed for ER [5]. While supervised techniques generally result in much better matching quality, these techniques require training data in the form of labeled true matching and true non-matching record pairs. In most practical applications of ER actual truth data are nonexistent or difficult to obtain, and training data have to be manually generated, a task known to be difficult both in terms of cost and quality [1], [2], [22].

As databases grow in size, the process of comparing record pairs becomes more challenging because the number of possible pairs grows quadratically with the size of the databases to be matched. Blocking and indexing techniques [5] are commonly applied to reduce the number of comparisons by splitting the databases into blocks and only comparing records within each block. Even after blocking, however, the number of generated *candidate record pairs* is often still much larger than the number of true matches, and therefore ER classification is generally a very imbalanced problem, which makes random sampling for selecting training data a serious challenge.

Several active learning techniques have been proposed for selecting training data for ER [1], [2], [6], [17], [19], [22]. The central idea of active learning is to reduce the labeling efforts through *actively* choosing *informative* or *representative* examples. Recently proposed active learning approaches [1], [6] for ER are heavily grounded on a monotonicity assumption: a record pair with higher overall similarity is more likely to be a true match than a pair with lower similarity. This assumption does not generally hold, as illustrated in Figure 1 which shows the distributions of true matches and non-matches from four real-world data sets where all four clearly violate the monotonicity assumption. This leads to the question of how can we effectively select training data if the monotonicity assumption does not hold and assuming the classes are imbalanced?

We propose an approach that, under a limited budget, exploits the cluster structure in data through active learning [7]. Our approach selects a subset (cluster) of informative weight vectors into the training data set by recursively splitting the set of weight vectors into smaller subsets until subsets are found that are *pure* (as we formally define in Section III). Pure subsets are those where the majority of their weight vectors refer to either matches or non-matches. The optimal number of examples to be labeled is calculated adaptively based on a sampling error margin. The resulting training set can then be used for any supervised ER classifier. Unlike other work, our approach does not rely on a monotonicity assumption.

We conduct an evaluation on four real-world data sets and compare our approach with several fully supervised and unsupervised ER classifiers, and with a recently proposed state-of-the-art active learning technique for ER [2]. Our results show that our approach can achieve classification performance comparable to fully supervised approaches with only a few hundred manual labels required for all four data sets.

II. Preliminaries and Problem Statement

Let $\mathbf{R}$ be a set of records from one or more data sets, each $r \in \mathbf{R}$ having a set of attributes $\mathbf{A}$. We use $r.a$ to refer to the value of an attribute $a \in \mathbf{A}$ in record r. Given two records $r_1, r_2 \in \mathbf{R}$, a similarity weight w of attribute a between r_1 and r_2, denoted as $w = f(r_1.a, r_2.a)$, is a value in the range $0 \leq w \leq 1$, where f is a function that quantifies the similarity

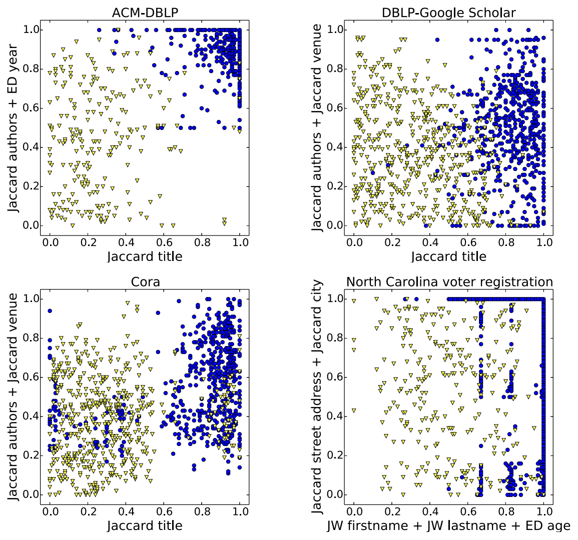

Fig. 1. Examples where the monotonicity assumption of similarities does not hold: Weight vectors that are generated from the data sets we use in our experiments (Section IV), with true matches shown as blue circles and true non-matches as yellow triangles. ED refers to the edit-distance and JW to the Jaro-Winkler approximate string comparison functions [5]. Similarities over several attributes are summed and normalized into $[0, 1]$.

between $r_1.a$ and $r_2.a$, such as an approximate string comparator [5]. For a set $\mathbf{A} = \{a_1, \ldots, a_d\}$ of attributes selected for performing ER tasks, each compared pair of records (r_1, r_2) results in a *weight vector* $\langle w_1, \ldots, w_d \rangle \in [0, 1]^d$, where w_i is the similarity weight of attribute a_i between r_1 and r_2.

A *weight vector set* $\mathbf{W}$ consists of all weight vectors over $\mathbf{A}$ which correspond to the pairs of two different records in $\mathbf{R}$ (potentially after blocking or indexing has been applied [5]). 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 of clusters $\{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$ that contains all weight vectors in $\mathbf{W}$, with $\mathbf{W}_i \cap \mathbf{W}_j = \emptyset$ for $1 \leq i \neq j \leq m$ and $\cup \mathbf{W}_i = \mathbf{W}$ for $1 \leq i \leq m$.

In practical applications of ER, human experts who are tasked with manual labeling of training examples often have different levels of expertise [5]. As a result, the accuracy of labeling will likely vary. Additionally, the cost of labeling in real-world applications is often restricted by operational constraints such as limited time or budget. To account for these practical limitations of manual labeling, we consider that a human oracle can be noisy [8], which simulates a non-perfect manual labeling process, and that it is budget-limited. Formally, a *human oracle* ζ over a weight vector set $\mathbf{W}_i$ is a function $\zeta : \mathbf{W}_i \mapsto \{M, N\}$, where M and N indicate the *match* and *non-match* status of a weight vector, respectively. ζ takes a set of weight vectors as input, and based on manual inspection of the attribute values of their corresponding records assigns each weight vector with either M or N. ζ is also associated with a pair $\langle bud(\zeta), acc(\zeta) \rangle$, where $bud(\zeta) > 0$ is a budget limit indicating the maximum total number, b_{tot}, of weight vectors that can be labeled by ζ, and $acc(\zeta) \in [0, 1]$ indicates the accuracy of the labels provided by ζ.

We view an *ER classifier* as a black-box that classifies record pairs into the two classes of matches, M, and non-matches, N, through their corresponding weight vectors [5]. 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. Once trained on $\mathbf{W}_i^T$, it classifies the weight vectors in $\mathbf{W}_i \setminus \mathbf{W}_i^T$ into the two subsets (clusters) $\mathbf{W}_i^M$ of matches and $\mathbf{W}_i^N$ of non-matches, with $\mathbf{W}_i^M \cap \mathbf{W}_i^N = \emptyset$. Any binary classifier, as previously used for ER classification [5], can be employed in our approach.

Analogously, an *ER selector* in our work is also considered as a black-box that, given a weight vector set $\mathbf{W}_i$ as input, selects and returns a subset of $\mathbf{W}_i$. The selection process should return a set of weight vectors that are informative for the set $\mathbf{W}_i$ and represent its characteristics. This selection can be based on different characteristics of $\mathbf{W}_i$, such as its quality, diversity, size, or location. We discuss several selection strategies for such an ER selector in Section III-B4.

Our interactive training data selection algorithm employs an ER classifier L and an ER selector S, and generates a set of clusters. Of these clusters, some are labeled as *pure* (as defined in Section III-B2) if the majority (according to a user set parameter) of weight vectors in a cluster are either in class M or N. Otherwise, a cluster is labeled as *fuzzy*. We use $\theta(L, S)$ to denote such a training data selection algorithm.

The *ER training data selection* problem we consider is to develop an algorithm $\theta(L, S)$ that generates a high quality training data set under a given budget constraint $bud(\zeta)$, i.e., $C_\theta(L, S) \leq bud(\zeta)$, where $C_\theta(L, S)$ is the labeling cost of $\theta(L, S)$ referring to the number of labeled weight vectors used in the training data selection process.

III. Interactive Training Data Set Selection

We next describe our algorithm in detail and then explain its key components in Section III-B.

A. Algorithm Description

The detailed description of our algorithm is presented in Algorithm 1. The input to the algorithm is a weight vector set $\mathbf{W}$, a set of functions and parameter settings, some of which will be described in Section III-B. The output of the algorithm are the final match and non-match training sets $\mathbf{T}^M$ and $\mathbf{T}^N$, with $\mathbf{T}^M \cap \mathbf{T}^N = \emptyset$, as selected from the given input weight vector set $\mathbf{W}$ (with $\mathbf{T}^M \subset \mathbf{W}$ and $\mathbf{T}^N \subset \mathbf{W}$).

The main iteration of the algorithm (line 3 onwards) loops as long as there are clusters in the queue $\mathbf{Q}$ and the total oracle budget b_{tot} has not been fully used ($b \leq b_{tot}$). In each iteration, the first cluster in the queue, $\mathbf{W}_i$, is processed depending upon CLUSTER_ORDER(). In line 5, based on the match proportion π_i we calculate n_i, the number of examples required for manual labeling for the selected cluster, as we will discuss in Section III-B1. In the first iteration (with $b = 0$ indicating no manual labeling has been done, lines 6 and 7), 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 subsequent iterations (lines 8 and 9) the MAIN_SELECT() function is used. Both types of selection functions select n_i informative weight vectors $\mathbf{S}_i$ from a cluster $\mathbf{W}_i$.

The selected weight vectors in $\mathbf{S}_i$ are then manually classified by the human oracle ORACLE() (line 10) into a match set $\mathbf{T}_i^M$ and a non-match set $\mathbf{T}_i^N$. These are both added to the final training sets in line 11, and removed from the cluster $\mathbf{W}_i$ in line 12. The used budget is also increased in line 12 by the number of manually classified weight vectors $|\mathbf{S}_i|$ (n_i).

Algorithm 1: ***Adaptive and Interactive Training Data Set Selection for Entity Resolution (AdInTDS)***

Input:
- Weight vector set: $\mathbf{W}$
- Budget limit: b_{tot}
- Minimum purity threshold: p_{min}
- Sampling margin of error: e
- Minimum size of a cluster: s_{min}
- Cluster ordering function: CLUSTER_ORDER() (as described in Section III-B3)
- Initial selection function: INIT_SELECT() (as described in Section III-B4)
- Main selection function: MAIN_SELECT() (as described in Section III-B4)
- Human oracle for labeling: ORACLE()
- Classifier function used for splitting clusters: CLASSIFIER()

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

```
1:  $\mathbf{T}^M = \emptyset; \mathbf{T}^N = \emptyset; b = 0$      // Initialize training sets as empty, and initialize number of so far manually labeled examples
2:  $\mathbf{Q} = [(\mathbf{W}, 0.5)]$      // Initialize queue of clusters with all weight vectors and match proportion estimate $\pi_i = 0.5$
3:  while $\mathbf{Q} \neq \emptyset$ and $b \leq b_{tot}$ do:      // As long as there are clusters in the queue and the budget has not been fully used
4:      $(\mathbf{W}_i, \pi_i) = $ CLUSTER_ORDER$(\mathbf{Q})$      // Get next cluster and its match proportion (according to queue ordering as described in Section III-B3)
5:      $n_i = z^2_{\alpha/2}\pi_i(1-\pi_i)/e^2$      // Calculate number of examples to select for manual labeling from cluster using Equation (1)
6:      if $b = 0$ then:      // First iteration (no manual labeling has so far been done)
7:          $\mathbf{S}_i = $ INIT_SELECT$(\mathbf{W}_i, n_i)$      // Initial selection of informative weight vectors, as discussed in Section III-B4
8:      else:      // Following iterations
9:          $\mathbf{S}_i = $ MAIN_SELECT$(\mathbf{W}_i, n_i)$      // Select informative weight vectors in later iterations, as discussed in Section III-B4
10:     $\mathbf{T}^M_i, \mathbf{T}^N_i, p_i = $ ORACLE$(\mathbf{S}_i)$      // Manually label selected weight vectors and calculate purity using Equation (2)
11:     $\mathbf{T}^M = \mathbf{T}^M \cup \mathbf{T}^M_i$; $\mathbf{T}^N = \mathbf{T}^N \cup \mathbf{T}^N_i$      // Add manually labeled weight vectors to training sets
12:     $\mathbf{W}_i = \mathbf{W}_i \setminus \mathbf{S}_i$; $b = b + |\mathbf{S}_i|$      // Remove manually labeled weight vectors from cluster, and update number of manual labels done so far
13:     if $p_i \geq p_{min}$ then:      // Cluster is pure enough
14:         if $|\mathbf{T}^M_i| > |\mathbf{T}^N_i|$ then:      // More matches than non-matches
15:             $\mathbf{T}^M = \mathbf{T}^M \cup \mathbf{W}_i$      // Add whole cluster to match training set
16:         else:      // More non-matches than matches
17:             $\mathbf{T}^N = \mathbf{T}^N \cup \mathbf{W}_i$      // Add whole cluster to non-match training set
18:     else if $|\mathbf{W}_i| > s_{min}$ and $b \leq b_{tot}$ then:      // Cluster has low purity and is large enough, and budget is not exhausted, so split cluster further
19:         if $\mathbf{T}^M_i \neq \emptyset$ and $\mathbf{T}^N_i \neq \emptyset$ then:      // Cluster contains both matches and non-matches
20:             CLASSIFIER.$train(\mathbf{T}^M_i, \mathbf{T}^N_i)$      // Train classifier on manually labeled weight vectors
21:             $\mathbf{W}^M_i, \mathbf{W}^N_i = $ CLASSIFIER.$classify(\mathbf{W}_i)$      // Classify current cluster, split into match and non-match sub-clusters
22:             $\mathbf{Q}.append((\mathbf{W}^M_i, p_i))$; $\mathbf{Q}.append((\mathbf{W}^N_i, p_i))$      // Append new clusters together with purity of parent cluster to queue
23: return $\mathbf{T}^M$ and $\mathbf{T}^N$
```

Besides the match and non-match sets, in line 10 the oracle also returns the purity p_i of the cluster, as will be described further in Section III-B2. If the purity of the cluster is equal to or higher than the minimum required cluster purity p_{min} (i.e. $p_i \geq p_{min}$), then all weight vectors in the cluster are added into one of the training sets (lines 14 to 17). On the other hand, if the purity of the cluster is not high enough, the cluster will be split further if (1) it is larger than the specified minimum cluster size s_{min}, (2) the total oracle budget b_{tot} has not been fully used, and (3) both training sets $\mathbf{T}^M_i$ and $\mathbf{T}^N_i$ are not empty (lines 18 to 21). If $\mathbf{T}^M_i$ and $\mathbf{T}^N_i$ are both not empty, then a classifier is trained using these two training sets (line 20), and this classifier is used to split the cluster $\mathbf{W}_i$ into a match and a non-match sub-cluster $\mathbf{W}^M_i$ and $\mathbf{W}^N_i$. These two sub-clusters are appended to the queue in line 22, together with the purity p_i of the parent cluster which is used as an estimate of the match proportion in these two sub-clusters.

B. Key Algorithm Components

1) Sample Size Calculation: Because of the limited oracle budget we aim to minimize the number of examples that need to be selected for manual labeling per cluster while ensuring there are enough examples to make informed decisions with high confidence. The generally high class imbalance in weight vector sets requires us to take the estimated proportion of matches in a cluster into account when calculating the number of samples required for a cluster of a certain size.

The required minimum sample size n for a certain confidence level for a given estimated proportion of matches π of the form $\pi \pm e$ (where the error margin e is specified by the user) can be calculated as [15] (line 5 in Algorithm 1):

$$n = \frac{z^2_{\alpha/2}\pi(1-\pi)}{e^2}, \qquad (1)$$

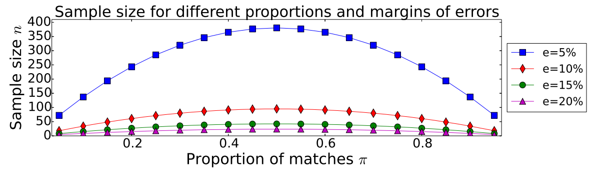

Fig. 2. Sample size n required for different estimates of the proportion of matches π and margins of error e for a 95% confidence level.

where $z_{\alpha/2}$ is the critical value of the Normal distribution at $\alpha/2$. For example, for a confidence level of 95%, $\alpha = 0.05$ and $z_{\alpha/2} = 1.96$ [15].

Figure 2 illustrates the sample size n for different proportions of matches π and different error margins e. As can be seen, with larger values of π (i.e. class imbalances) smaller sample sizes are required, while for smaller error margins e the sample size increases significantly. In our algorithm, in the first iteration we set $\pi = 0.5$ (line 2) as we do not know the actual class imbalance, while in all following iterations we set π as the purity of a cluster's parent cluster as this provides us with an estimate of a cluster's class imbalance (line 22).

2) Cluster Purity: We calculate the *purity* p_i of a cluster $\mathbf{W}_i$ based on the labeling of the human oracle (line 10) using the manually labeled weight vector set $\mathbf{S}_i$ as the proportion of labeled weight vectors that have the majority label:

$$p_i = purity(\mathbf{W}_i) = max\left(\frac{m_i}{m_i + u_i}, \frac{u_i}{m_i + u_i}\right), \qquad (2)$$

where $m_i = |\mathbf{T}^M_i|$ and $u_i = |\mathbf{T}^N_i|$ with $m_i + u_i = |\mathbf{S}_i|$. For a given minimum purity threshold $p_{min} \in [0.5, 1]$, $\mathbf{W}_i$ is said to be *pure* if $purity(\mathbf{W}_i) \geq p_{min}$; otherwise $\mathbf{W}_i$ is *fuzzy*. As we split a weight vector set $\mathbf{W}_i$ into smaller clusters (line 21 in the algorithm), the purity of the sub-clusters will increase if the accuracy of the classifier is better than 50%, because the classifier will split $\mathbf{W}_i$ such that matches are more likely in sub-cluster $\mathbf{W}_i^M$ and non-matches in $\mathbf{W}_i^N$.

3) Cluster Queue Ordering: With a limited budget, the human oracle can only label a restricted number (b_{tot}) of weight vectors. The cluster $\mathbf{W_i}$ selected from the queue $\mathbf{Q}$ for manual labeling at each iteration (line 4 of Algorithm 1), therefore, should be the one that can provide the best improvement in the quality and coverage of the training data. We consider the following CLUSTER_ORDER() methods:

FI-FO: In this method clusters are processed in the order they are appended to $\mathbf{Q}$ (first-in first-out) in line 22. This means no cluster characteristics are considered in the ordering [22].

MaxPuri: In each iteration this method selects the cluster with the highest purity, as calculated using Equation 2 for its parent cluster (line 10) because the purity of newly created clusters (in line 21) is unknown and could only be established by manual classification. Our hypothesis is that the cluster with the highest purity will more likely achieve the p_{min} threshold compared to clusters with lower purity, and given a limited budget this cluster is therefore more likely added into the training set than other clusters.

MinPuri: In this method we select the cluster with the lowest purity based on the purity of its parent. Our hypothesis is that clusters with low purity are the most ambiguous ones, and in line with traditional active learning approaches for ER [17], [19], splitting such ambiguous clusters to obtain smaller clusters will likely lead to some of the most difficult to classify weight vectors to be manually labeled.

Close01: This method follows the ***01Init*** initial selection method, where clusters closest to the $[1]^d$ (exact match) and $[0]^d$ (totally different) corners are selected. Following earlier work [3], our hypothesis is that clusters closest to these corners will more likely be pure and therefore added to the training data set. We calculate the centroid distance (average link) for each cluster in $\mathbf{Q}$ to $[1]^d$ and $[0]^d$ and select the cluster with the smallest distance to either of these.

CloseMid: In this method, we select the cluster farthest away from both the $[1]^d$ and $[0]^d$ corners. Similar to the ***MinPuri*** method, our hypothesis is that this selects clusters for labeling that are most ambiguous, leading to a training set that contains difficult to classify weight vectors.

Balanced: In this method we consider both the characteristics of the clusters in $\mathbf{Q}$ and the current size of the training data sets $\mathbf{T}^M$ and $\mathbf{T}^N$. We follow the ***Close01*** method, but select the cluster closest to either the $[1]^d$ or $[0]^d$ corner such that the resulting sizes of $\mathbf{T}^M$ and $\mathbf{T}^N$ become more balanced [12].

Sample: In this method, we select the cluster which has the largest ratio of cluster size divided by the number of samples required, i.e. $argmax_i(|\mathbf{W}_i|/n_i)$. Our hypothesis is that this cluster will give us the best splitting of the weight vector space for the smallest manual labeling effort (n_i).

Weighted: In this final method, we weight different criteria to get an overall score for each cluster. Specifically, we calculate $s_i = w_s \cdot ncs_i + w_p \cdot (p_i - 0.5)/0.5 + w_l \cdot (1 - nss_i) + w_b \cdot s_b$ for each cluster in the queue $\mathbf{Q}$, where w_s, w_p, w_l, and w_b are the weights for cluster size, cluster purity, cluster labeling sample size, and training set balance, respectively, with $w_s + w_p + w_l + w_b = 1$. ncs_i is the normalized cluster size for $\mathbf{W}_i$ in the queue $\mathbf{Q}$ with regard to all clusters in $\mathbf{Q}$, and similarly nss_i is the normalized sample size of $\mathbf{W}_i$. The balance score is calculated as $s_b = (|\mathbf{T}^M|/(|\mathbf{T}^M| + |\mathbf{T}^N|) - \pi_i)$ This balance score will be large for clusters which lead to a more balanced training set.

4) Weight Vector Selection: The informativeness of the weight vectors selected for manual labeling crucially influences the quality of the final training data set. The selection functions used in Algorithm 1 (lines 7 and 9) need to be chosen carefully. The aim of these functions is to select weight vectors from a cluster $\mathbf{W}_i$ into $\mathbf{S}_i$ that are able to represent the match and non-match status of all weight vectors in $\mathbf{W}_i$.

Following the selection methods evaluated in [9], to represent the diversity of the weight vectors in a cluster, we propose three methods for INIT_SELECT() (line 7 in Algorithm 1):

FarInit: Using the farthest-first clustering algorithm [11], this method selects the n_i (sample size) weight vectors in $\mathbf{W}_i$ that are farthest apart from each other. The hypothesis is to start with a set of vectors with the highest possible diversity.

01Init: Two records that have the same values in all d compared attributes, leading to a weight vector $o = [1]^d$, likely refer to the same entity. Two records that have completely different values in all attributes, with a weight vector $z = [0]^d$, very likely refer to two different entities. Following earlier work on training data selection for ER [3], we initialize the training set by selecting the $n_i/2$ weight vectors in $\mathbf{W}_i$ closest to o and the $n_i/2$ vectors closest to z.

CorInit: This approach combines the ideas of both previous methods by selecting weight vectors with the highest possible diversity in terms of all attributes in $\mathbf{A}$. We select the n_i weight vectors that are closest to the corners of the d-dimensional hypercube of similarity values $\{[v_1, \ldots, v_d] | v_i \in \{0, 1\}$ for $i = 1, \ldots, d\}$, where there are 2^d corners in total. This method requires $n_i \geq k2^d$ in order to select at least k weight vectors per corner.

For the MAIN_SELECT() function used in all iterations after the first (line 9 in Algorithm 1) we consider the following two methods to select weight vectors from a cluster $\mathbf{W}_i$:

Far: Similar to the ***FarInit*** method, this method selects weight vectors from $\mathbf{W}_i$ with the largest distances between each other using the farthest-first clustering algorithm [11].

FarMed: This method first uses the ***Far*** method to select $n_i - 1$ farthest apart weight vectors in $\mathbf{W}_i$, and then selects the medoid weight vector closest to the center of the cluster. The hypothesis behind this approach is to obtain a better picture of the distribution of matches and non-matches in a cluster by considering both the boundary and the center of the cluster.

IV. Experiments and Results

We conducted experiments on four data sets, as summarized in Table I: ACM-DBLP and DBLP-Google Scholar (DBLP-GS) [13], CORA[1], and a North Carolina Voter Registration (NCVR) database[2]. We used the *Febrl* system for pairwise comparisons [4]. For each data set we employed several blocking keys and various string comparison functions [5]. The output of this step are sets of weight vectors of the compared record pairs and their true match and non-match labels.

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

TABLE I. CHARACTERISTICS OF DATA SETS USED IN EXPERIMENTS.

Data set name(s)	Number of records	Number of unique weight vectors	$M : N$ class imbalance	Time for pair-wise comparisons	Attributes used for blocking / Dimensionality of weight vectors
ACM-DBLP [13]	2,616 / 2,294	687,910	1 : 1785	79.2 sec	Title, venue, authors, year / 3
DBLP-GS [13]	2,616 / 64,263	8,124,258	1 : 3273	868.3 sec	Title, venue, authors, year / 3
CORA[1]	1,295	286,141	1 : 16	47.3 sec	Title, venue, authors, publisher, year / 4
NCVR[2]	224,073 / 224,061	3,495,580	1 : 27	363.6 sec	First/last name, street, zip code, city / 5

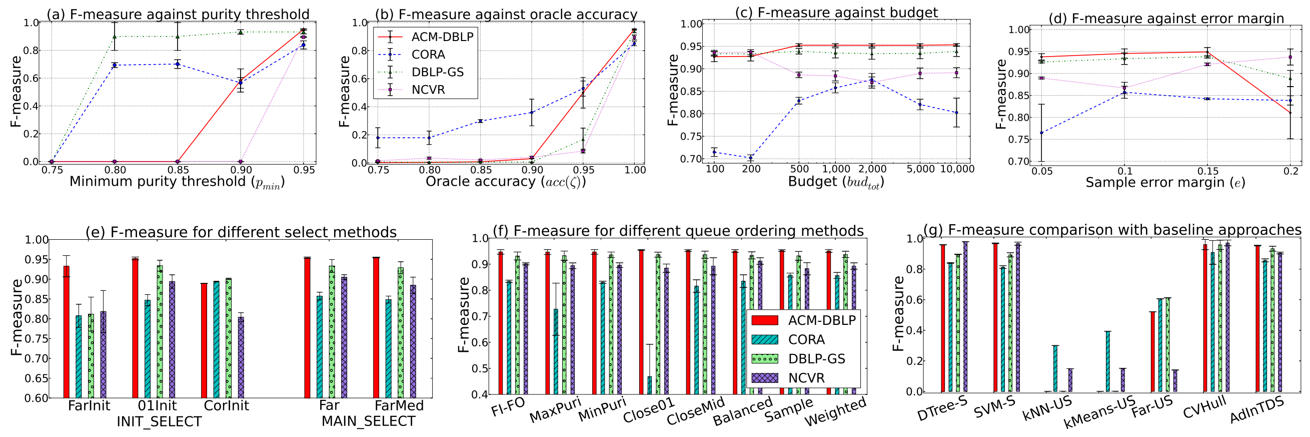

Fig. 3. F-measure against (a) minimum purity threshold, (b) oracle accuracy, (c) budget, (d) sampling error margin, (e) different initial and main selection methods, (f) different queue ordering methods, and (g) comparison with baseline approaches. Note the different y-axis scales.

We used the following parameter variations in our experiments: minimum purity 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, **1000**, 2000, 5000, 10000], sample error margin e = [0.05, **0.1**, 0.15, 0.2], and all selection and cluster queue ordering methods discussed above. Default parameter values, that gave us the best results based on an extensive set of initial experiments, are shown in bold font. The classifier used for splitting weight vectors (lines 20 and 21 in Algorithm 1) was a decision tree with either entropy or information gain [16] as attribute selection measure.

We evaluated the effectiveness of our approach using F-measure [5]. We compared our approach (which we refer to as ***AdInTDS***) with: (1) fully supervised decision tree (***DTree-S***), (2) fully supervised SVM with linear and polynomial kernels (***SVM-S***), (3) unsupervised k-nearest neighbor clustering (***kNN-US***) [3], (4) unsupervised k-means clustering (***kMeans-US***), and (5) unsupervised farthest-first clustering (***Far-US***) [4]. We also compared our approach with a state-of-the-art active learning approach for ER as proposed by Bellare et al. [2] (***CVHull***). Our approach and all baseline approaches were implemented in Python 2.7.3 (except ***CVHull*** which was implemented in C++), and we ran all experiments on a Ubuntu 14.04 server with 2.4 GHz CPUs and 128 GBytes of memory. The programs and data sets are available from the authors.

We investigated how different values for the main parameters of our approach (i.e., p_{min}, $acc(\zeta)$, b_{tot}, and e) and the INIT_SELECT(), MAIN_SELECT() and CLUSTER_ORDER() methods determine the quality of the ER classification results. As shown in Figure 3 (a), F-measure increases with larger minimum purity thresholds since higher minimum purity leads to clusters of higher quality, which will improve the classification results. As expected, the F-measure also gets better when the accuracy of the oracle increases (Figure 3 (b)), where only a perfect labeling process leads to high quality results. These two parameters, minimum purity and oracle accuracy, are the ones which influence the performance of our approach the most.

Larger budgets (b_{tot}) allow more weight vectors to be manually labeled, resulting in small increments of F-measure results, as shown in Figure 3 (c). An interesting aspect of our approach is that an F-measure ≥ 0.8 is achieved on all data sets even with a small budget of $b_{tot} = 500$. The sample error margin results, shown in Figure 3 (d), indicate that our approach is quite robust with regard to this parameter.

The F-measure results for different selection methods are presented in Figure 3 (e). The three initial selection methods perform well with average F-measure of 0.8 or above on all four data sets. The two main selection methods ***Far*** and ***FarMed*** also perform similarly. In Figure 3 (f) we show how different orderings of clusters to be selected for labeling affect the quality of the generated training data set. The ***Balanced***, ***Sample*** and ***Weighted*** methods outperform the other methods in terms of higher F-measure on most data sets.

Finally, we compared our approach with the six baseline approaches described above. The F-measure results in Figure 3 (g) illustrate that our approach can achieve significantly higher results compared to unsupervised approaches, and results comparable to the fully supervised approaches and to ***CVHull***. While ***CVHull*** achieved very high F-measure results on all data sets (however with quite large variations), this approach does not consider a limited budget and generally required a larger number of manual labels to achieve its high results.

In Table II, we show F-measure results and corresponding budget requirements of our approach compared with published results of two other active learning approaches for ER [2], [10] on the DBLP-GS and CORA data sets. As can be seen, our approach performs comparatively and it achieves higher or similar F-measure results for a smaller budget.

TABLE II. COMPARISON OF F-MEASURE RESULTS FOR CERTAIN BUDGETS WITH OTHER PUBLISHED ACTIVE LEARNING ER WORK.

Method	Budget	F-measure	Budget	F-measure
	DBLP-GS data set [13]		ACM-DBLP data set [13]	
Bellare et al. (CVHull) [2]	860–940	0.92	400–470	0.965
Gokhale et al. [10]	2,082	0.921	–	–
AdInTDS	200	0.938	200	0.938
AdInTDS	500	0.945	500	0.956
AdInTDS	1,000	0.946	1,000	0.957

V. RELATED WORK

Early work on active learning for ER was based on a committee of classifiers that were iteratively refined using the most ambiguous examples for labeling [17], [19]. An active selection method of similar pairs was investigated in [12] to select a balanced number of matches and non-matches.

More recent work [1], [2] has concentrated on the objective of learning quality, such that given a minimum precision specified by the user, a classifier was learned to achieve a precision greater than this minimum and a recall close to the best possible. Efficiency of active learning for ER has been addressed using two techniques: (1) incorporating blocking or indexing [5] and (2) optimizing active learning algorithms under certain distribution assumptions, such as monotonicity of similarities [1], [6] or low noise in the data [2]. A hierarchical clustering approach for active learning, and the bias introduced by this approach, have been discussed in [7].

A graph-based active name disambiguation technique was proposed in [23] where the aim was to identify which name mentions within sets of documents refer to the same person. A similar approach is CrowdER [20]. In both approaches, a small number of highly similar pairs were used to train an initial classifier, followed by the manual labeling of ambiguous pairs by a crowd (non-experts) system. Neither of these approaches however considered a limited budget nor imperfect labeling, which have to be expected when using crowd-sourcing.

Two approaches that consider noisy labels and budget constraints for selecting examples to be manually labeled in crowd-sourced applications have been proposed in [14]. A strategy for managing the limited labeling budget was proposed in [21] to ask questions in decreasing order of pair-wise similarities. Several instance selection strategies for active learning in imbalanced class problems have been evaluated in a recent study [9], where the authors found that no selection method outperformed all others on different data sets.

A number of studies have attempted to control labeling noise [8]. Repeated labeling strategies were investigated in [18], while [24] proposed an approach to select the most reliable oracle among multiple noisy oracles for labeling.

VI. CONCLUSIONS AND FUTURE WORK

We have presented an adaptive and interactive training data selection approach for ER that significantly reduces labeling costs while generating training data, and that can achieve high matching quality comparable to fully supervised training.

While crowd-sourcing has shown to provide powerful solutions for ER [10], [20], [23], many applications of ER deal with the matching of sensitive data (like medical records) where privacy and confidentiality concerns prohibit the use of crowd services for manual labeling. Therefore, approaches such as ours will continue to play an important role for efficiently generating high quality training data for ER.

As future work, we plan to incorporate the accuracy of an oracle into the calculation of purity as well as the number of samples required, and to develop ways to jointly optimize the selection and ordering problems with the aim to maximize the quality of training data generated by our approach.

VII. ACKNOWLEDGMENTS

This research was partially funded by the Australian Research Council, Veda, and Funnelback Pty. Ltd., under LP100200079. We like to thank Kedar Bellare for providing the code for ***CVHull***, and Khoi-Nguyen Tran for assistance.

REFERENCES

[1] ARASU, A., GÖTZ, M., AND KAUSHIK, R. On active learning of record matching packages. In *ACM SIGMOD* (2010).

[2] BELLARE, K., IYENGAR, S., PARAMESWARAN, A. G., AND RASTOGI, V. Active sampling for entity matching. In *ACM SIGKDD* (2012).

[3] CHRISTEN, P. Automatic training example selection for scalable unsupervised record linkage. In *PAKDD* (2008).

[4] CHRISTEN, P. Development and user experiences of an open source data cleaning, deduplication and record linkage system. *SIGKDD Explorations 11*, 1 (2009).

[5] CHRISTEN, P. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer, 2012.

[6] DAL BIANCO, G., GALANTE, R., GONCALVES, M., CANUTO, S., AND HEUSER, C. A practical and effective sampling selection strategy for large scale deduplication. *IEEE TKDE 27*, 9 (2015), 2305–2319.

[7] DASGUPTA, S., AND HSU, D. Hierarchical sampling for active learning. In *IEEE ICML* (2008).

[8] DU, J., AND LING, C. Active learning with human-like noisy oracle. In *IEEE ICDM* (2010).

[9] FERDOWSI, Z., GHANI, R., AND SETTIMI, R. Online active learning with imbalanced classes. In *IEEE ICDM* (2013).

[10] GOKHALE, C., DAS, S., DOAN, A., NAUGHTON, J., ET AL. Corleone: Hands-off crowdsourcing for entity matching. In *ACM SIGMOD* (2014).

[11] HOCHBAUM, D. S., AND SHMOYS, D. A best possible heuristic for the k-center problem. *MOR 10*, 2 (1985).

[12] KÖPCKE, H., AND RAHM, E. Training selection for tuning entity matching. In *VLDB QDB/MUD* (2008).

[13] KÖPCKE, H., THOR, A., AND RAHM, E. Evaluation of entity resolution approaches on real-world match problems. *PVLDB 3*, 1-2 (2010).

[14] MOZAFARI, B., SARKAR, P., FRANKLIN, M., JORDAN, M., AND MADDEN, S. Scaling up crowd-sourcing to very large datasets: A case for active learning. *PVLDB 8*, 2 (2014).

[15] OTT, R. L., AND LONGNECKER, M. T. *An Introduction to Statistical Methods and Data Analysis*, 6th ed. Brooks/Cole, 2010.

[16] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., AND ET AL. Scikit-learn: Machine learning in Python. *JMLR 12* (2011).

[17] SARAWAGI, S., AND BHAMIDIPATY, A. Interactive deduplication using active learning. In *ACM SIGKDD* (2002).

[18] SHENG, V. S., PROVOST, F., AND IPEIROTIS, P. G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *ACM SIGKDD* (2008).

[19] TEJADA, S., KNOBLOCK, C., AND MINTON, S. Learning domain-independent string transformation weights for high accuracy object identification. In *ACM SIGKDD* (2002).

[20] WANG, J., KRASKA, T., FRANKLIN, M. J., AND FENG, J. CrowdER: Crowdsourcing entity resolution. *PVLDB 5*, 11 (2012).

[21] WANG, J., LI, G., KRASKA, T., FRANKLIN, M., ET AL. Leveraging transitive relations for crowdsourced joins. In *ACM SIGMOD* (2013).

[22] WANG, Q., VATSALAN, D., AND CHRISTEN, P. Efficient interactive training selection for large-scale entity resolution. In *PAKDD* (2015).

[23] WANG, X., TANG, J., CHENG, H., AND YU, P. S. Adana: Active name disambiguation. In *IEEE ICDM* (2011).

[24] WU, W., LIU, Y., GUO, M., WANG, C., AND LIU, X. A probabilistic model of active learning with multiple noisy oracles. *Neurocomputing 118* (2013), 253–262.