Active Sampling for Entity Matching with Guarantees

KEDAR BELLARE, Facebook Inc.
SURESH IYENGAR, Microsoft Research Lab India
ADITYA PARAMESWARAN, Stanford University
VIBHOR RASTOGI, Google Inc.

In entity matching, a fundamental issue while training a classifier to label pairs of entities as either duplicates or non-duplicates is the one of selecting informative training examples. Although active learning presents an attractive solution to this problem, previous approaches minimize the misclassification rate (0-1 loss) of the classifier, which is an unsuitable metric for entity matching due to class imbalance (i.e., many more non-duplicate pairs than duplicate pairs). To address this, a recent paper [Arasu et al. 2010] proposes to maximize recall of the classifier under the constraint that its precision should be greater than a specified threshold. However, the proposed technique requires the labels of all n input pairs in the worst-case.

Our main result is an active learning algorithm that approximately maximizes recall of the classifier while respecting a precision constraint with provably sub-linear label complexity (under certain distributional assumptions). Our algorithm uses as a black-box any active learning module that minimizes 0-1 loss. We show that label complexity of our algorithm is at most $\log n$ times the label complexity of the black-box, and also bound the difference in the recall of classifier learnt by our algorithm and the recall of the optimal classifier satisfying the precision constraint. We provide an empirical evaluation of our algorithm on several real-world matching data sets that demonstrates the effectiveness of our approach.

Categories and Subject Descriptors: H.2.8 [**Information Systems**]: Database Management—*Data Mining*; H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Clustering*

Additional Key Words and Phrases: entity matching, deduplication, active learning, imbalanced data

ACM Reference Format:
Bellare, K., Iyengar, S., Parameswaran, A., and Rastogi, V. 2012. Active Sampling for Entity Matching with Guarantees. ACM Trans. Embedd. Comput. Syst. 9, 4, Article 39 (March 2010), 23 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Entity Matching (EM) is the problem of determining if two entities in a dataset refer to the same real-world object. Entity matching is a complex and ubiquitous problem that appears in numerous application domains (including image processing, information extraction and integration, and natural language processing), often under different terminology (e.g., coreference resolution, record linkage, and deduplication [Elmagarmid et al. 2007]).

Machine learning approaches [Winkler 1999; Chaudhuri et al. 2007; Bilenko and Mooney 2003a] for entity matching often learn a classifier over pairs of entities labeling them as duplicate or non-duplicate. The classifier is built over models such

---

A large fraction of this work was done when the authors were employed at Yahoo! Research, Inc.
Author's addresses: K. Bellare, Facebook Inc., 1601 Willow Road, Menlo Park, CA, USA; S. Iyengar, Microsoft Research Lab India Pvt. Ltd. "Vigyan", #9, Lavelle Road, Bangalore, India; A. Parameswaran, Stanford University, 353 Serra Mall, Stanford, CA, USA; V. Rastogi, Google Inc., Mountain View, CA, USA.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

as SVM or logistic regression, using features like string similarity. The models are trained over labeled example pairs, which are usually costly to obtain. Active learning techniques [Dasgupta and Langford 2009; Karampatziakis and Langford 2011; Balcan et al. 2009; Hanneke 2007; Beygelzimer et al. 2010; Beygelzimer et al. 2009] are therefore used to carefully select the examples to label while learning a good classifier — one that minimizes the 0-1 loss. These algorithms provide label complexity guarantees, i.e., guarantees on the number of labeled examples required to learn the classifier.

In entity matching, the number of duplicate pairs is typically small, $o(m)$, for a dataset of m entities. On the other hand, the total number of pairs including non-duplicates can be $\Theta(m^2)$. While blocking techniques [Whang et al. 2009; McCallum et al. 2000] are used to reduce the number of pairs considered for entity matching from $\Theta(m^2)$ to a more manageable number, denoted n, the non-duplicates still vastly outnumber the duplicate pairs, often by a ratio of 100 to 1, as evident in many benchmark entity matching datasets. In such cases, minimizing 0-1 loss is insufficient: a classifier that labels all pairs as non-duplicates has a small 0-1 loss of less than 1%. Thus even this extremely poor classifier is considered good under 0-1 loss.

To address this, a recent paper [Arasu et al. 2010] considers precision and recall[1] of the classifier as the objective functions. Precision in entity matching is the fraction of pairs labeled as duplicates that are true duplicates, and recall is the fraction of true duplicates labeled as duplicates. So the classifier that labels all pairs as non-duplicates, labels no true duplicates as duplicates, and has a recall of 0. Thus the classifier is correctly considered poor under the recall metric.

Arasu et al. [Arasu et al. 2010] provide an active learning algorithm to maximize recall under a constraint that precision should be greater than a specified threshold. The algorithm makes a monotonicity assumption on the precision and recall of the classifier over its parameter space. The algorithm then searches for the optimal classifier essentially through a binary search over classifiers in the high-dimensional parameter space. Due to this high-dimensional binary search, the algorithm has high label complexity (i.e., many labeled examples requested) and high computational complexity, and in the worst-case requires the labels of all n input pairs. Furthermore, the monotonicity assumption does not usually hold in practice, and while the algorithm still returns a feasible classifier satisfying the precision constraint, its recall is often poor.

In this paper, we propose an active learning algorithm for optimizing recall under the precision constraint, using as a black-box any active learning approach that minimizes 0-1 loss. We first use a Lagrange multiplier to cast the precision constrained recall optimization problem into an unconstrained one with an objective that is a linear combination of precision and recall. For a fixed value of the Lagrange multipier, the linear objective can be optimized using the given black-box. Then we search for the right value of the Lagrange multiplier. For this we embed all classifiers in a two dimensional space, and perform a search along the convex hull [Preparata and Shamos 1985] of the embedded classifiers.

A major challenge in our work is to ensure that the search over the embedded classifiers is efficient, and we show that via discretization techniques, we are able to achieve a rather low number of worst-case $O(\log^2 n)$ calls to the black-box. We also show that additional calls would not help: our output classifier is guaranteed to be the best one that can be found using any number of calls to the black-box. By comparison, [Arasu et al. 2010] has exponential (in the number of dimensions) worst-case computational complexity.

[1]That work actually considers an approximation of recall metric, but we ignore this distinction for now.

While the classifier output by our algorithm need not be optimal in terms of its recall, we show that it is pareto-optimal (i.e., no other classifier dominates it in both recall and precision). We also provide additional guarantees on how close our output classifier is to the optimal by showing that they lie on the same edge of the convex hull of the embedded classifiers, and coincide in case the optimal is a vertex point.

Contributions and Outline:

- Our main result is an active learning algorithm that approximately maximizes recall of the classifier under the precision constraint using as a black-box any active learning approach that minimizes 0-1 loss. We show that label complexity of our algorithm is at most $O(\log^2 n)$ times the label complexity of the black-box. We also show that the classifier learnt by our algorithm is pareto-optimal, and close to the true optimal in terms of recall.
- We use the IWAL active learning algorithm [Beygelzimer et al. 2010] as the black-box for 0-1 loss minimization. IWAL has been shown to have a sublinear label complexity under certain low noise assumptions. Since our algorithm has a label complexity of at most $O(\log^2 n)$ times that of IWAL, it also has sublinear label complexity under the same low noise assumptions. To our knowledge, this is the first active learning algorithm with sublinear label complexity for optimizing the precision-constrained recall objective.
- We provide an empirical evaluation of our algorithm on several real-world matching datasets and compare it with the technique in [Arasu et al. 2010]. We show that our algorithm achieves better quality over these datasets while significantly reducing label and computational complexity.

We present background on active learning in Sec. 2, followed by our approach in Sec. 3. We then discuss experimental evaluation in Sec. 5 and conclude in Sec. 6.

2. BACKGROUND

In this section, we define the notation and setup the problem formally. We also briefly review the important active learning techniques.

2.1. Notation

Let E be the set of all entities for the matching task, and m be the size of E. We assume that the set of all candidate entity pairs to be matched from E has been generated (say by some blocking technique) and denote it as C. We denote $n = |C|$ be the number of candidate pairs. For a pair of entities (e_1, e_2), we assume that a d-dimensional feature vector $x = (x_1, x_2, \ldots, x_d)$ represents the similarity between entity e_1 and e_2, and let X be the set of all feature vectors corresponding to pairs in C. We say that the label of $x \in X$, denoted $y(x)$, is 1 if x corresponds to a duplicate pair, and -1 otherwise. In this paper, we consider linear classifiers defined as follows.

Definition 2.1. A linear classifier h is represented by a d-dimensional vector $w = (w_1, \ldots, w_d)$. h classifies a feature vector x as positive, i.e. $h(x) = 1$ if $w.x \geq 0$ and negative, i.e. $h(x) = -1$, otherwise.

We let H be the set of all linear classifiers. For any classifier $h \in H$, we define number of false positives, false negatives, true positives as the following:

$$fp(h) = \sum_{x \in X} \mathbf{1}(h(x) = 1 \wedge y(x) = -1)$$
$$fn(h) = \sum_{x \in X} \mathbf{1}(h(x) = -1 \wedge y(x) = 1)$$
$$tp(h) = \sum_{x \in X} \mathbf{1}(h(x) = 1 \wedge y(x) = 1)$$

The precision and recall of a classifier are then defined as $\text{precison}(h) = \frac{tp(h)}{tp(h)+fp(h)}$ and $\text{recall}(h) = \frac{tp(h)}{tp(h)+fn(h)}$.

Problem. We wish to maximize recall under the precision constraint. Formally, this is stated as below.

PROBLEM 1 (RECALL). *Given* $\tau \in [0, 1]$, *find* $h \in H$ *to*

$$\textbf{maximize} \quad \textit{recall}(h)$$
$$\textbf{subject to} \quad \textit{precison}(h) \geq \tau$$

The RECALL problem is difficult to solve because recall(h) and precison(h) are complex functions of h.

2.2. Active Learning

Now we briefly review the main problems and techniques in active learning.

Problems. In active learning, the focus has primarily been to solve the problem of minimizing 0-1-loss. We call this problem as the 01-LOSS problem. The goal of the problem[2] is to minimize the total number of false negatives and false positives. We state it below.

PROBLEM 2 (01-LOSS). *Find* $h \in H$ *to*

$$\textbf{minimize} \ \frac{fn(h) + fp(h)}{n}$$

A slight generalization of the problem considers a weighted loss function in which false negatives and false positives have different penalties. We denote the problem as 01-LOSSWEIGHTED and define it below.

PROBLEM 3 (01-LOSSWEIGHTED). *Given* $\alpha \in [0, 1]$, *find* $h \in H$ *to*

$$\textbf{minimize} \ \frac{\alpha fn(h) + (1-\alpha) fp(h)}{n}$$

Techniques. One of the first active learning algorithms was given in [Cohn et al. 1994]. The algorithm solves the 01-LOSS problem for *separable* datasets – datasets for which a classifier exists having 0 empirical 0-1 loss. The paper showed that a good classifier can be learnt for separable datasets using only $O(\log n)$ labeled examples. The

[2]Learning literature considers a more difficult version of 01-LOSS problem when the examples come from an unknown distribution, here we consider the simpler problem using the empirical distribution, for which the same techniques apply.

algorithm was based on the idea of selective sampling: each example point is queried with a probability, computed based on the point and previously labeled examples.

Since [Cohn et al. 1994], several approaches have been proposed to handle non-separable datasets. Most recently, IWAL [Beygelzimer et al. 2010] proposed an efficient active learning algorithm having sublinear label complexity under some distributional assumptions. The algorithm is again based on selective sampling, and assigns a probability of querying an example based on the disagreement between two classifiers, each learnt on all previously labeled examples, but differing in the labels for the example of interest. The algorithm guarantees the following properties.

THEOREM 2.2 (IWAL [BEYGELZIMER ET AL. 2010]). *IWAL approximately minimizes 01-loss using $O(error(h^*)n\theta + \sqrt{n \log n}\theta)$ labeled examples, where $error(h^*)$ is the 01-loss of the optimal classifier, and θ, the disagreement coefficient [Hanneke 2007], is a parameter depending on the source distribution and the hypothesis class, but independent of the number of points n.*

Under certain distributional assumptions [Beygelzimer et al. 2010], θ is a small constant and $error(h^*) = o(1)$. Thus the label complexity is provably sublinear under those assumptions.

3. OUR LEARNING ALGORITHMS

In this section, we describe our approach to solve the RECALL problem given a black-box to solve the 01-LOSS problem. We use two algorithms: (i) CONVEXHULL algorithm that approximately solves RECALL using a solution for the 01-LOSSWEIGHTED problem, and (ii) REJECTIONSAMPLING algorithm that reduces an instance of 01-LOSSWEIGHTED problem into that of the 01-LOSS problem. We describe each of the algorithms below. But we first begin by slightly generalizing the RECALL problem.

Problem Generalization: The RECALL problem maximizes recall under the precision constraint. Maximizing recall(h) of a classifier h is equivalent to minimizing the fraction of false negatives, $fn(h)/n$. (Note that $tp(h) + fn(h)$ is a constant independent of h.) Instead of just minimizing this fraction, we consider here a more general problem that minimizes a linear combination of false negatives and false positives, $\frac{\alpha fn(h)+(1-\alpha)fp(h)}{n}$, under the precision constraint.

Further, the constraint on precision can be transformed into a simpler one. Denoting $\epsilon = (1-\tau)/\tau$, we can write.

$$\frac{tp(h)}{tp(h)+fp(h)} \geq \tau \equiv \epsilon \cdot tp(h) - fp(h) \geq 0$$

This gives us a generalized version of the RECALL problem that we call as RECALL-WEIGHTED.

PROBLEM 4 (RECALLWEIGHTED). *Given $\alpha \in [0,1]$ and $\epsilon \in [0,\infty)$, find $h \in H$ to*

$$\begin{aligned} \textbf{minimize} \quad & \tfrac{\alpha fn(h)+(1-\alpha)fp(h)}{n} \\ \textbf{subject to} \quad & \epsilon \cdot tp(h) - fp(h) \geq 0 \end{aligned}$$

Note that for $\alpha = 1$, the solution for RECALLWEIGHTED problem is same as that of RECALL problem.

3.1. The CONVEXHULL Algorithm

Now we describe the CONVEXHULL algorithm that approximately solves the RECALL-WEIGHTED problem by repeatedly solving the 01-LOSSWEIGHTED problem. Note that the objective of both problems is the same, but RECALLWEIGHTED has an additional

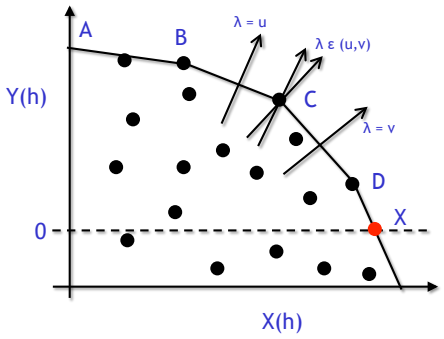


Fig. 1: Convex Hull Search

precision constraint. The CONVEXHULL algorithm essentially removes the constraint using a trick similar to Lagrange multipliers. We describe this below.

Embedding. We embed a classifier h as a point in two dimensional space with the first coordinate equal to negative of the objective and the second to the slack of the precision constraint, as shown below.

$$X(h) = \frac{-(\alpha fn(h) + (1-\alpha) fp(h))}{n} \tag{1}$$

$$Y(h) = \frac{\epsilon \cdot tp(h) - fp(h)}{n} \tag{2}$$

Thus, the RECALLWEIGHTED problem is equivalent to finding a classifier h that has the highest $X(h)$ under the constraint that $Y(h) \geq 0$. Note also that finding a classifier h that maximizes $X(h) + \lambda Y(h)$ for a given λ can be shown to be equivalent to the 01-LOSSWEIGHTED problem (with appropriate α).

Let $P = \{(X(h), Y(h)) : h \in H\}$ be the set of all two-dimensional embeddings of all possible linear classifiers. While P need not be a convex set, we denote C to be the convex hull polytope of points in P and say a classifier h lies on C if $(X(h), Y(h))$ lies on an edge of C. Any classifier h lying on C is pareto-optimal: no other classifier h' exists that has both $-X(h')$ (i.e. objective) as well as $Y(h')$ (i.e. precision) better than h. We will show that CONVEXHULL algorithm returns classifiers on the convex hull. (Hence the name.)

Figure 1 shows how the convex hull polytope may look like in a specific scenario. As can be seen in the figure, each dot indicates the embedding of (one or more) classifiers in terms of $(X(h), Y(h))$ pairs. The classifiers (i.e., dots) on the boundary of the convex hull polytope correspond to the the pareto-optimal classifiers. For instance, the classifiers that embed to points `A`, `B` and `C` are all pareto-optimal. We desire classifiers that have $Y(h) \geq 0$. Thus, in the figure, we desire classifiers that embed to points on or above the dashed line. The best classifier in H for our problem would be one that maps

ALGORITHM 1: CONVEXHULL Algorithm

Input: A set of n pairs X,
Black-box $B(\lambda)$ returning h with max $X(h) + \lambda Y(h)$
Output: Approximate solution for $RecallWeighted$
Compute sorted list Λ of all values λ, s.t. $-1/\lambda$ is a line slope in the embedded space.
Set $min = 0, max = |\Lambda|$.
while $min < max$ **do**
  Set $mid = (min + max)/2$
  Set λ to be $\Lambda[mid]$
  $h = B(\lambda)$
  if $Y(h) \geq 0$ set $max = mid$, else set $min = mid$
end while
return h

to point X, i.e., the point at which the line $Y(h) = 0$ intersects the convex polytope. Note that there may not be a classifier which maps to that point.

Even though the set of all linear classifiers H is exponentially large in the number of dimensions, the size of P is bounded by $O(n^3)$, since many classifiers embed to the same point in P. To see this, note that embedded coordinates $X(h)$ and $Y(h)$ are functions of $fn(h)$, $fp(h)$, and $tp(h)$, each of which can vary from 0 to n. Thus, the number of dots in the two-dimensional plane is polynomial in n, even though multiple classifiers may embed to the same dot.

Since H is exponentially large, and P much smaller, we perform our search for optimal classifier in the embedded space. For the search, we define S the set of all possible slopes of lines joining any two points in P, and Λ a sorted array of all possible values λ where $-1/\lambda$ is a slope in S. The slopes in S have the form $\frac{Y(h')-Y(h)}{X(h')-X(h)}$, and it is easy to show that both S and Λ have at most $O(n^3)$ distinct values.

The CONVEXHULL **Algorithm.** We first describe the algorithm and provide some intuition as to why it works. Later, we prove desirable properties of the algorithm.

For the algorithm, we assume a black-box B for the 01-LOSSWEIGHTED problem that takes a λ and returns a h maximizing $X(h) + \lambda Y(h)$.

When we use our black box B with a specific λ, we get a classifier corresponding to a point on the convex hull C. So, one approach to find a good classifier is to use the black box B with every $\lambda \in \Lambda$, get the optimal classifier for that λ, and return the classifier that has the highest $X(h)$, with $Y(h) \geq 0$. This approach is inefficient, since Λ can be fairly large, specifically, $O(n^3)$. Next, we describe an efficient approach that utilizes more fine-grained properties of how the black box B interacts with the convex hull.

On applying the black box B to find a classifier with the largest $X(h) + \lambda Y(h)$, we obtain a classifier corresponding either to a point on the line in C with slope $-1/\lambda$, or to a vertex in C when $-1/\lambda$ is between the slopes of two adjoining lines in C. To clarify this point, let the slope of CD in Figure 1 be $-1/v$, and the slope of BC be $-1/u$. Then, if we apply the black box B with $\lambda = v$, then we get a classifier on line CD, and if we apply B with $\lambda = u$, then we get a classifier on line BC. For any $\lambda \in (u, v)$, we get a classifier corresponding to vertex C.

Since Λ contains all slopes between points in P, it also contains all slopes between neighboring points in the convex hull C. Moreover, the slopes of lines on the convex hull, as well as $Y(h)$, monotonically decrease as $X(h)$ increases. Thus, we can simply apply binary search over the values of Λ to find a classifier with the smallest $Y(h)$, where $Y(h) \geq 0$. This classifier will be the one closest to X on the convex hull, and yet satisfies $Y(h) \geq 0$.

The algorithm, shown in Algorithm 1, essentially does a binary search over values in Λ. Starting from the two extreme elements $\Lambda[min], \Lambda[max]$, the algorithm repeatedly picks their midpoint $\lambda = \Lambda[(max + min)/2]$, and then computes a classifier h maximizing $X(h) + \lambda Y(h)$ using the black-box B. Depending on whether $Y(h) \geq 0$, the intervals are updated to ensure that we always have at least one extreme point that satisfies feasibility requirement $Y(h) \geq 0$. Finally the algorithm terminates with a feasible solution, i.e. a classifier h with $Y(h) \geq 0$.

Note however, that since we only get *some* point along any specific line in C, we may not be able to get the best classifer. In fact, the best guarantee that we can get is that instead of getting a classifier that maps to X, we may get a classifier that maps to D. (Note there definitely is such a classifier, because all vertices of the convex hull correspond to classifiers.)

More formally, we can show the following additional properties of the algorithm.

Complexity: First, we show that the algorithm takes a logarithmic number of steps to find a classifier.

THEOREM 3.1 (COMPLEXITY OF CONVEXHULL). *The* CONVEXHULL *algorithm terminates with a feasible classifier (i.e. classifier satisfying the precision constraint) with label complexity at most* $3 \log n$ *times that of black-box* B *used in the algorithm.*

PROOF. Since size of Λ is bounded by $O(n^3)$, the binary search procedure finishes in at most $3 \log n$ iterations. This immediately gives the required label and computational complexity bounds for the algorithm. The feasibility of the solution is guaranteed as a result of the binary search as long as one classifier h exists with $Y(h) \geq 0$, i.e satisfies the precision constraint. This is always true, for a linear classifier labeling all points as negative. □

On Optimality: In general, the solution returned by the CONVEXHULL algorithm need not be optimal. Recall the set C, the convex hull polytope of embedded points. Each classifier on C is pareto-optimal: i.e. no other classifier dominates it on both the value of objective and precision. The optimal classifer is guaranteed to lie on the convex hull, while the algorithm can return some other point on the same edge. Many times the two coincide, and the algorithm does return the optimal classifier. We state this result formally below.

THEOREM 3.2. *The optimal solution lies on the convex hull* C. *The* CONVEXHULL *algorithm returns a classifier on the same edge of the convex hull. If the optimal is a vertex point of C, then the algorithm returns the optimal solution.*

PROOF. First note that any pareto-optimal classifier lies on the convex hull. If not, then extending the line $Y = X$ from the point towards the convex hull results in a classifier dominating it in both objective value as well as precision. Since optimal is also pareto-optimal, it has to lie on the convex hull.

Next we show that the black-box $B(\lambda)$ also returns a h on the convex hull C. Assume the contrary, and suppose it returns a classifier h not on the convex hull. Then for h, there is a point on an edge AB of the convex hull that strictly dominates it in terms of both objective value and precision. Then one of the two vertices, either A or B, has as good or better $X + \lambda Y$ value, contradicting the assumption.

Furthermore, we show that $h = B(\lambda)$ lies on an edge AB, then its slope has to be $-1/\lambda$. This follows directly from linearity of the objective function $X + \lambda Y$. Thus our black box function gives us, for each λ, either a vertex of the convex hull, or a point on the line of the convex hull that has slope $-1/\lambda$. Let λ_0 be the smallest λ for which $h = B(\lambda)$ is feasible, i.e. $Y(h) \geq 0$. If $-1/\lambda_0$ is a slope whose value lies between slopes of two existing edges in the convex hull, then the binary search of the CONVEXHULL

algorithm would give the vertex corresponding to λ_0. Furthermore, that vertex will be optimal. If $-1/\lambda_0$ is the slope of one of the edges (say AB) of the convex hull, then the binary search find λ_0 and $B(\lambda_0)$ will return a point on AB. Also the optimal will be on the same edge. □

On Efficiency: As stated, Algorithm 1 needs to explicitly construct the sorted list Λ that can have $O(n^3)$ elements. Thus the overall time complexity of the algorithm is $O(n^3) + 3\log n \cdot T(B)$: $O(n^3)$ during construction of Λ and $3\log n \cdot T(B)$ during active learning, where $T(B)$ is the time complexity of black-box B used in the algorithm.

Note that the $O(n^3)$ term might make the algorithm infeasible for large datasets. We now show how to eliminate the explicit construction of Λ, when α and ϵ are restricted to fractions of fixed bit length b (thus encompassing representation of floats in modern computer processors.) Our approach is to design a $\Lambda' \supseteq \Lambda$ whose constituent values are equally spaced apart (at a discretization factor γ). We may then simply do a binary search over the integral multiples of γ and avoid explicit enumeration of Λ'.

Since α and ϵ are fractions of bit length at most b, we can show using Equations 1 and 2 that numerators of $X(h)$ and $Y(h)$ are an integral multiple of $1/2^b$. Now, consider the smallest difference between two slopes in Λ. Let r/s and t/u be two slopes. We have: $r/s - t/u = (ru - ts)/su$. First, the smallest non-negative value of $ru - ts$ can be shown to be $1/(2^{2b}n^2)$. Second, the value of su is at most 1. Thus, the smallest difference between slopes is $1/(2^{2b}n^2)$. We use this value as our discretization factor γ. The largest value of $\lambda \in \Lambda$ is $O(n^3)$, and thus, the new size of Λ' is $O(n^3)/\gamma = O(n^5 2^{2b})$. Now, since Λ' contains all values from 0 to n^3 in multiples of $1/(2^{2b}n^2)$, we may perform binary search without explicitly enumerating the values in Λ'. Thus, the overall time complexity is $(5\log n + 2b\log 2) \cdot T(B)$.

3.2. The REJECTIONSAMPLING Algorithm

In this section we describe the REJECTIONSAMPLING algorithm that reduces the 01-LOSSWEIGHTED problem into an instance of 01-LOSS problem. Recall that the difference in the two problems is that while the former minimizes any linear combination, $\frac{\alpha fn(h)+(1-\alpha)fp(h)}{n}$, of the false negatives and false positives, the latter only minimizes their sum. Both the problems are however unconstrained. Our algorithm is very similar to idea of rejection sampling used for transforming cost-sensitive binary classification into the standard setting [Zadrozny et al. 2012; Mineiro 2003]. Our analysis is slightly different as we use empirical loss functions rather than distribution ones.

Black-box: We assume a black-box B for solving the 01-LOSS problem. We further assume that B reads all input points one by one, maintaining some internal state as it is reading the points, which it uses to determine whether or not to ask the label for the next point. We model this behavior as follows: B accepts as input a sequence of points $\bar{x} = x_1, \ldots, x_k$ along with their labels $\bar{y} = y_1, \ldots, y_k$, and a new point x, and returns *true* or *false* indicating whether or not to query the label for this point. Once all points have been considered, a function B_h accepts all the points $\bar{x}$ for which labels were asked, along with all their labels $\bar{y}$, and returns the classifier h minimizing 01-loss. Note that the label complexity of the black-box is exactly the size of sequence $\bar{x}$, since those are points that the black-box decided to query.

Since the 01-LOSSWEIGHTED problem has a penalty α for false negative and $1-\alpha$ for false positive, one can use B for solving the 01-LOSSWEIGHTED problem if the distribution of false negatives and false positives is changed appropriately. This can be done simply by rejecting a positive point with probability $1-\alpha$ and a negative point with probability α. However, given just a point we do not know its label, and thus cannot decide its rejection probability. To overcome this we query B to check whether

ALGORITHM 2: REJECTIONSAMPLING Algorithm

Input: A set of n points X,
A black-box B for 01-LOSS problem
Output: A solution for 01-LOSSWEIGHTED problem
Let $\bar{x} = (), \bar{y} = ()$ be empty sequences.
for each point x in X **do**
  If $B(\bar{x}, \bar{y}, x)$ = FALSE, continue to next point.
  Otherwise, query the point x. Let y be its label
  Toss a coin with success probability α if $y = 1$ and $1 - \alpha$ if $y = -1$
  If the coin returns failure, continue to next point
  Otherwise, add x and y to $\bar{x}$ and $\bar{y}$ respectively.
end for
return $B_h(\bar{x}, \bar{y})$

or not query the point's label, and if it returns *true*, we use the label to set the rejection probability. This procedure works, but since we are rejecting even labeled points, the label complexity takes a hit.

The algorithm formally corresponding to the above intuition is defined in Algorithm 2. It begins with empty sequences $\bar{x}$, $\bar{y}$ for points and their labels. It then repeatedly picks a new point, and uses the black-box to determine whether or not to query the point's label. The black-box is provided all points $\bar{x}$ and their labels $\bar{y}$. If the black-box decides not to query the point, it is ignored and the algorithm moves to the next point. Otherwise, the points label is determined, and used to determine the rejection probability. If the point is rejected, then it is ignored even though its label has already been queried. Otherwise, the point is added to the sequence of labeled points $\bar{x}$. Finally, the black-box is fed all the labeled points in $\bar{x}$ and their labels, and the returned classifier is output.

Since REJECTIONSAMPLING algorithm is randomized, we can only show probability bounds on its optimality and label complexity. The following theorem shows that w.h.p the objective value of the returned classifier is within $O(1/\sqrt{n})$ away from the optimal. Additionally, it shows that label complexity is bounded in expectation. The same label complexity bound can be shown to hold w.h.p.

THEOREM 3.3. *Let h_{rs} be solution returned by the* REJECTIONSAMPLING *algorithm. Then with probability at least $1 - \delta$, the difference in the objective value of h_{rs} and the optimal is at most $O(\sqrt{\log \delta / n})$. Furthermore, the expected label complexity of the algorithm is at most $max(\frac{\alpha}{1-\alpha}, \frac{1-\alpha}{\alpha})$ times the label complexity of the black-box B for the* 01-LOSS *problem.*

PROOF. Denote $FN(h), FP(h)$ the set of false negatives and false positives for a classifier h. Since some false positives and false negatives are rejected in a random run of the REJECTIONSAMPLING algorithm, denote for a point x, $r(x)$ the random variable equal to 1 if x is selected in REJECTIONSAMPLING, and 0 if it is rejected.

The black-box B essentially minimizes the objective $Obj(h)$ equal to $\sum_{x:FN(h)} r(x)/n + \sum_{x':FP(h)} r(x')/n$. Also expectation $E(h)$ of $Obj(h)$ over the coin tosses of the algorithm is simply $(\alpha fn(h) + (1 - \alpha) fp(h))/n$, which is the right objective for the 01-LOSSWEIGHTED problem. Using Hoeffding, we can say that w.p. $1 - \delta$, $Obj(h)$ deviates from its expectation $E(h)$, by at most $O(\sqrt{\log \delta / n})$.

This shows that If h^* is an optimal classifier for the 01-LOSSWEIGHTED problem, w.p. $1 - \delta$, $|Obj(h^*) - E(Obj(h^*)|$ is bounded by $O(\sqrt{\log \delta / n})$. Similarly, w.p. $1 - \delta$, $|Obj(h_{rs}) - E(Obj(h_{rs})|$ is bounded by $O(\sqrt{\log \delta / n})$. Now we know that since the black-

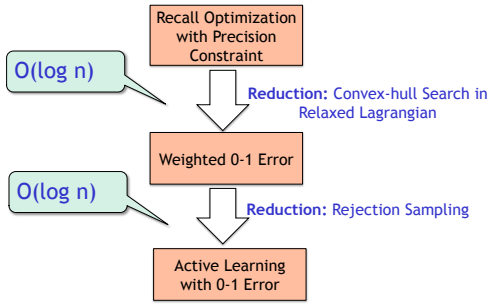


Fig. 2: Outline

box returned h_{rs}, $Obj(h_{rs}) \leq Obj(h^*)$. Since h^* is optimal for the 01-LOSSWEIGHTED problem, and $E(h*)$ the objective for the same, we know $E(h*) \leq E(h_{rs})$. This shows that difference in $Obj(h_{rs})$ and $E(h*)$ is no more than $O(\sqrt{\log \delta/n})$. Hence proved.

The proof of expected label complexity follows directly from the expected number of labeled examples rejected by the REJECTIONSAMPLING algorithm. □

3.3. Overall Approach

Now we describe our overall approach, depicted in Figure 2. We do not introduce new techniques here, but explain and analyze how the various algorithms are run together. We begin by running the CONVEXHULL algorithm for solving the RECALL problem. During its run, CONVEXHULL might make calls to its black-box for solving the 01-LOSSWEIGHTED problem. Whenever a call is made, we invoke a run of the REJECTIONSAMPLING algorithm. When that in turn makes a call to its black-box for solving the 01-LOSS problem, we invoke a run of the IWAL algorithm [Beygelzimer et al. 2010]. We describe this process in detail below. We also compute the label complexity of the overall process.

Run of CONVEXHULL algorithm: We use CONVEXHULL algorithm to solve an instance of the RECALLWEIGHTED problem with $\alpha = 1 - 1/\log n$. Note that the objective that we are optimizing is then $\frac{(1-1/\log n)fn(h)+1/\log n fp(h)}{n}$, instead of $\frac{fn(n)}{n}$ that we need to do for solving the RECALL problem. However, since $\alpha = 1 - 1/\log n$ is very close to 1, the two objectives can be shown to be at most $1/\log n$ away from each other. Thus an approximate solution for RECALLWEIGHTED for this α is a good approximate solution for the RECALL problem. Here we solved RECALLWEIGHTED instead of RECALL to ensure bounded labeled complexity when running the REJECTIONSAMPLING algorithm, which we discuss next.

Run of REJECTIONSAMPLING algorithm: During the above run of CONVEXHULL algorithm, whenever a call to the black-box for solving the 01-LOSSWEIGHTED problem is made, we invoke the REJECTIONSAMPLING algorithm. Note that the objective for

the 01-LOSSWEIGHTED problem is given by $X(h) + \lambda Y(h)$, which expands to

$$\frac{(1 - 1/\log n + \epsilon\lambda) fnh + (1/\log n + \lambda) fp(h)}{n}$$

Thus REJECTIONSAMPLING algorithm solves an instance of 01-LOSSWEIGHTED problem with

$$\alpha' = \frac{1 - 1/\log n + \epsilon\lambda}{1 + (1 + \epsilon)\lambda}$$

This α' determines the rejection probabilities and the label complexity of the REJECTIONSAMPLING algorithm. Finally, when the run is complete, we return the output classifier to the CONVEXHULL algorithm

Run of IWAL algorithm [Beygelzimer et al. 2010]**:** During the above run of REJECTIONSAMPLING algorithm, whenever a call to the black-box for solving the 01-LOSS problem is made, we invoke the IWAL algorithm, which is described in detail in [Beygelzimer et al. 2010]. We only use it as a black-box here, and return its output to the REJECTIONSAMPLING algorithm.

Now we reason about the total number of examples that are queried in the above process.

THEOREM 3.4 (OVERALL LABEL COMPLEXITY). *The label complexity of our overall approach is at most $O(\log^2 n)$ times that of the IWAL algorithm. Since IWAL algorithm has a label complexity of $O(error(h^*)n\theta + \sqrt{n \log n}\theta)$, the overall label complexity is $O(2 \log^2 n \left[error(h^*)n\theta + \sqrt{n \log n}\theta\right])$. Under certain distributional assumptions [Beygelzimer et al. 2010], this label complexity is sublinear.*

PROOF. From theorem 3.1, the label complexity of CONVEXHULL is $2 \log n$ times that of the black-box REJECTIONSAMPLING used in the algorithm. Further, since

$$\alpha' = \frac{1 - 1/\log n + \epsilon\lambda}{1 + (1 + \epsilon)\lambda} \text{ and } 1 - \alpha' = \frac{1/\log n + \lambda}{1 + (1 + \epsilon)\lambda},$$

we can show that

$$max(1/\alpha', 1/(1 - \alpha') \leq max(\log n, \frac{\epsilon}{1 + \epsilon}, \frac{1}{1 + \epsilon})$$

Since ϵ is a constant independent of n, $max(1/\alpha', 1/(1 - \alpha'))$ is $O(\log n)$. Thus from Theorem 3.3, the label complexity of REJECTIONSAMPLING is at most $O(\log n)$ times the label complexity of the black-box IWAL used. Finally, since the IWAL complexity is $O(error(h^*)n\theta + \sqrt{n \log n}\theta)$ as shown in Theorem 2.2, we get the required result. □

4. RELATED WORK

The work related to us can be placed under three categories. We describe each of them in turn.

Entity Matching. Many techniques have been proposed for the entity matching problem [Elmagarmid et al. 2007; Köpcke and Rahm 2010]. The ones most relevant to us are learning-based techniques that train a classifier over labeled pairs of examples. These include naive bayes [Winkler 1999], decision trees [Chaudhuri et al. 2007], SVMs [Bilenko and Mooney 2003a]. All of these techniques are fully supervised, i.e., they do not try to reduce the label complexity by choosing the pairs whose labels to request. To reduce the number of labeled examples, [Köpcke and Rahm 2008] describes an algorithm that uses a heuristic string similarity function and then samples pairs having varied similarity scores. The algorithm cannot directly be applied for active

learning as it is only for training data selection, but can in fact be used in conjunction with our active learning algorithm as a preprocessing step to select the pool of candidate pairs. Another training set construction method for deduplication is the described in [Bilenko and Mooney 2003b]. The training examples selected by this approach are a mixture of examples with high textual similarity and randomly sampled examples. In contrast to this static approach, our active learning algorithm is more dynamic and systematically explores the example space while also providing guarantees on precision.

Active learning for entity matching. Several authors have previously studied the application of active learning to entity deduplication problems [Christen 2005; Rendle and Schmidt-Thieme 2008; Sarawagi and Bhamidipaty 2002; Tejada et al. 2001]. However, these approaches do not provide any guarantees in terms of precision or recall, which is undesirable for datasets with highly imbalanced classes.

The previous work most similar to ours is [Arasu et al. 2010], which also provides an active learning algorithm to maximize recall under a constraint that precision should be greater than a specified threshold. Their algorithm essentially performs a binary search in the high-dimensional parameter space by assuming monotonicity in precision with respect to parameters. In the worst-case the algorithm queries the labels of all the examples and its computational complexity is exponential in the number of parameters. Our approach, in contrast, provides guarantees on precision while being computationally and label efficient. In addition, it does not require a monotonicity assumption which may not hold in general and therefore can be applied to other imbalanced classification problems.

Active learning for general classification. Our method is also related to active learning for classification tasks. There is a large and growing body of work [Settles 2009] on active learning for binary classification. A good summary of recent theoretical work in the area is provided in [Dasgupta and Langford 2009]. However, almost all related work on active learning for the general binary classification problem focuses on 0-1 loss minimization. We described two such related works in Section 2. In our work, we use IWAL [Beygelzimer et al. 2010] as a black-box active learner for 0-1 loss.

To our knowledge, [Beygelzimer et al. 2009] is the only active learning algorithm that minimizes general loss functions in addition to 0-1 loss. However, the algorithm cannot perform constrained optimization required for ensuring our precision constraint. Furthermore, no direct efficient implementation of algorithm is known for the class of linear classifiers. Previous approaches [Balcan et al. 2009; Dasgupta et al. 2007; Hanneke 2007] also require maintaining a candidate set of hypotheses (called a version space), which is computationally infeasible. Recent agnostic active learning approaches, which our algorithm builds upon, achieve sub-linear label complexity [Balcan et al. 2009; Beygelzimer et al. 2009; Beygelzimer et al. 2010; Hanneke 2007] when assumptions are made about low label noise in the data distribution. It is also known that under unbounded noise models, sub-linear label complexity is impossible [Balcan et al. 2007; Hanneke 2009]. An exponential improvement in label complexity for unbounded noise was shown recently but it assumes that data has multiple views [Wang and Zhou 2010].

The majority of active learning research has focused on querying the most informative examples needed to learn the classifier. More recent papers have also considered querying representative examples [Dasgupta and Hsu 2008], and querying examples that are both representative and informative [Huang et al. 2010].

5. EXPERIMENTS

In this section, we describe our experimental setup (Section 5.1) and present results comparing our approach against a previous state-of-the-art algorithm [Arasu et al. 2010] on several real-world datasets (Section 5.2). We then perform a more thorough analysis of our algorithm on varying parameters, on how quickly it progresses to the solution, and on varying the properties of the dataset (Section 5.3).

Dataset	Schema	Size	Ratio (+/-)	Blocking Function	Features
Business	name(N), street(S), city(C), zip(Z), phone(P)	3958	0.115	Jacc(N) ≥ 0.2, Jacc(S) ≥ 0.2, EQ(P)	5
Person	first-name(F), sex(S), birthdate(D), zip(Z), last-name(L)	574913	0.004	SoundEQ(F), EQ(S), SoundEQ(L), EQ(D)	9
DBLP-ACM	authors(A), title(T), venue(V), year(Y)	494437	0.005	Char3Jacc(T) ≥ 0.05	7
Scholar-DBLP	authors(A), title(T), venue(V), year(Y)	589326	0.009	Char3Jacc(T) ≥ 0.1	7

Table I: Description of datasets used in our experiments. The blocking functions used to reduce the number of pairwise comparisons during matching are Jaccard similarity (Jacc), string equality (EQ), phonetic equality (SoundEQ) and trigram Jaccard similarity (Char3Jacc). The number of similarity features is shown in the last column.

5.1. Setup

5.1.1. Datasets. A brief description of the four real-world datasets used in our experiments is shown in Table I. These include:

- Business: This is a dataset of local business listings used in the production system at Yahoo!. Each listing contain attributes like name, street, phone, etc. of the business. A set of labeled pairs is obtained as follows: we tokenize the name and street attributes into sets of k-grams. We then do blocking using a combination of prefix-filtering and min-hash to select pairs of entities that have a Jaccard overlap of more than 0.2 on the set of k-grams for either name or street attributes. Then using a heuristic matching function that gives a score to each pair, we select pairs having varied values for the scores, similar to the technique in [Köpcke and Rahm 2008]. Finally, the selected pairs are judged by human editors to generate a dataset of 3958 labeled pairs.
- Person: This is a record linkage dataset from the UCI machine learning repository [Frank and Asuncion 2010]. It contains 574,913 pairs for which features have been computed and labels have been provided.
- DBLP-ACM [Köpcke and Rahm 2008]: This is a large bibliography record linkage task in which 494,437 matching records in dblp and acm have been manually labeled as either duplicates or non-duplicates.
- Scholar-DBLP [Köpcke and Rahm 2008]: This is a dataset of 589,326 pairs similar to dblp-acm. It however includes automatically extracted records from google scholar, which cause data quality problems and make matching harder.

The last three datasets are highly imbalanced.

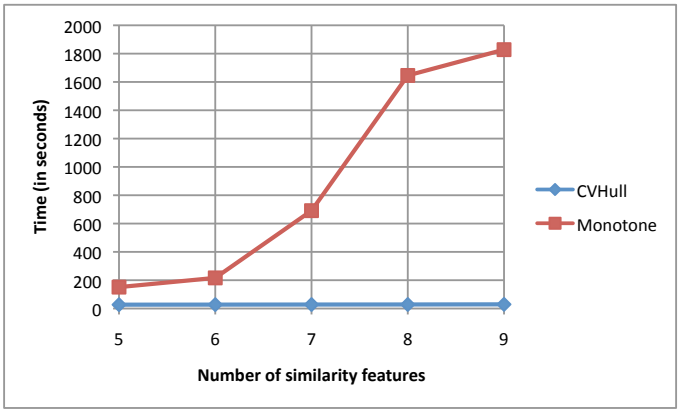


Fig. 3: The computational complexity of the two algorithms as the dimension is varied.

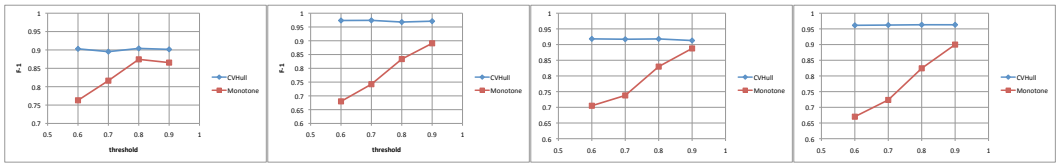
5.1.2. Algorithms. We implement three algorithms for active learning.

- MONOTONE: This is our implementation of the algorithm described in [Arasu et al. 2010]. The precision evaluations required for the algorithm is done over random samples of size 100. To ensure that the same examples be sampled for each precision evaluation whenever possible, a random permutation is chosen and fixed. Then each random sample is generated by selecting the first 100 applicable examples from this permutation.
- VW: This is an implementation of the IWAL algorithm [Beygelzimer et al. 2010] obtained from the open source Vowpal Wabbit [Karampatziakis and Langford 2011]. The algorithm uses stochastic gradient descent to learn a linear classifier. The 01-loss is approximated as squared-loss for efficient implementation.
- CVHULL: This is an implementation of both CONVEXHULL and REJECTIONSAMPLING algorithms on top of IWAL as described in Sec. 3.3. Whenever CONVEXHULL calls its black-box, an implementation of REJECTIONSAMPLING is invoked. When REJECTIONSAMPLING calls for its black-box, VW is invoked. The VW algorithm reads examples one at a time, which we feed to the algorithm in a randomly chosen order kept fixed across its different invokation. The precision constraint check required in CONVEXHULL is implemented by a precision evaluation of over random samples of size 100, generated in the same manner as in MONOTONE.

Note that all of our datasets consist of labeled examples. The active learning algorithms are run on the datasets as follows: as the active learning algorithm reads the examples the labels are kept hidden. Whenever the algorithm requires a label for a specific example, its label is read from the dataset and given to the algorithm. This cuts out the need for a human involvement of labeling during our experiments.

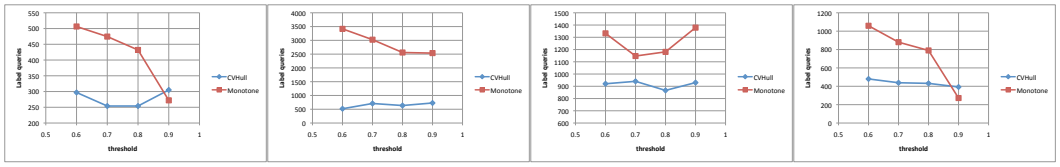
5.2. Experimental Comparison

In Fig. 4, we report the F-1 achieved by MONOTONE and CVHULL on all datasets, as the threshold used for the precision constraint is varied. We do not report the F-1 for VW, since it returns classifier that often violates the precision constraint. The graphs show than CVHULL has consistently higher F-1 that MONOTONE over all datasets and precision thresholds. The difference in F-1 between the two algorithms is as big as 0.15 in some cases, but becomes smaller (around 0.05 on average) for highest precision threshold of 0.9. While this difference is still significant, it does indicate that the gains



(a) Business (b) Person (c) Scholar-DBLP (d) DBLP-ACM

Fig. 4: The effect of varying the precision threshold on the F-measure for different datasets.



(a) Business (b) Person (c) Scholar-DBLP (d) DBLP-ACM

Fig. 5: The effect of varying the precision threshold on the number of label queries for different datasets.

at high precision thresholds are limited, possibly owing to limited choices of classifiers satisfying the precision constraint.

In Fig. 5, we report the number of queries required by MONOTONE and CVHULL on all datasets, as the threshold used for the precision constraint is varied. We again do not report the number for VW, since it returns a classifier that often violates the precision constraint. The graphs show that CVHULL requires significantly lower number of queries than MONOTONE on all but 2 cases. The difference in number of examples is particularly stark for Person and Scholar-DBLP datasets, sometimes more than 3000 examples. A particularly attractive property of CVHULL is that it learns the classifier in around 500 points for all but one datasets and all threshold values.

In Fig. 3, we compare the computation time for the two algorithms MONOTONE and CVHULL as the number of dimensions is varied for the Person dataset. The figure clearly demonstrates that MONOTONE has an exponential complexity w.r.t number of dimensions, while CVHULL has an almost a constant dependence.

In Fig. 6, we show that that VW often fails to produce a feasible classifier, i.e. one that satisfies the precision constraint. The graph plots the success rate, i.e. the fraction of times, over 10 random runs, the algorithm outputs a feasible classifier. Obviously as the precision threshold is increased, number of feasible classifiers decrease, and so does the success rate. This graph demonstrates the need for the CONVEXHULL and REJECTIONSAMPLING algorithms used in CVHULL on top of VW.

5.3. Additional Experiments

In this section, we describe additional experiments to address three separate concerns: (a) how the parameters of our algorithm affect its performance (b) the progress of our algorithm over iterations of the binary search (c) how the algorithm is impacted by properties of the dataset. These experiments will further validate the usefulness of our algorithms under different scenarios.

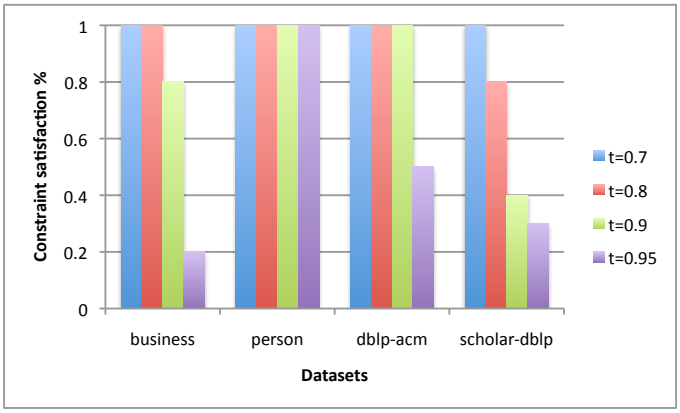


Fig. 6: The constraint satisfaction percentage of VW active learning algorithm over 10 random runs for different datasets.

5.3.1. Parameter Sensitivity Experiments. We begin by considering how sensitive the algorithms CVHULL and MONOTONE are to parameters that we arbitrarily fixed in the experiments.

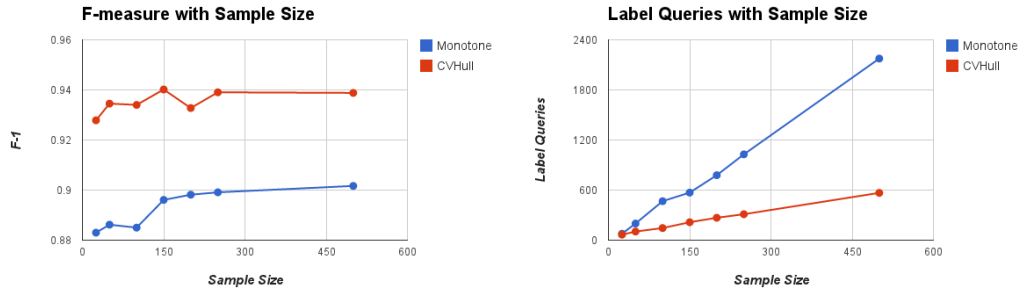
Precision Sampling: In the experiments in the previous section, we used a fixed sample of size 100 to do precision evaluation, both for MONOTONE and CVHULL. In this experiment, we would like to see how the two algorithms are affected on changing the sample size. We fixed the precision threshold at 0.85.

In Figure 7, we plot (a) the F-1 score and (b) label complexity of MONOTONE and CVHULL as a function of the sample size for the DBLP-ACM dataset, with the sample size varying between 30 and 500. As can be seen in the figure, for both algorithms, the F-1 score stabilizes (with small variations) very quickly, while the number of queries issued increases as a function of sample size. Moreover, for all sample sizes, CVHULL strictly outperforms MONOTONE in both F-1 and label complexity. In particular, note that the rate of increase of label complexity of MONOTONE is much higher than CVHULL. Note also that CVHULL achieves very good F-1 even for a very small sample size (30) — thus, in practice, we may prefer using small sample sizes resulting in a reasonable F-1 (0.93 — compared to the optimal 0.95) and very low label complexity (less than 100).

We repeated the experiments for the Person dataset and the results are shown in Figure 8. The results are similar, showing that very good performance is achieved by CVHULL for very small sample sizes, as low as 30, across various datasets. In this case, the label complexity of CVHULL increases very slowly compared to MONOTONE.

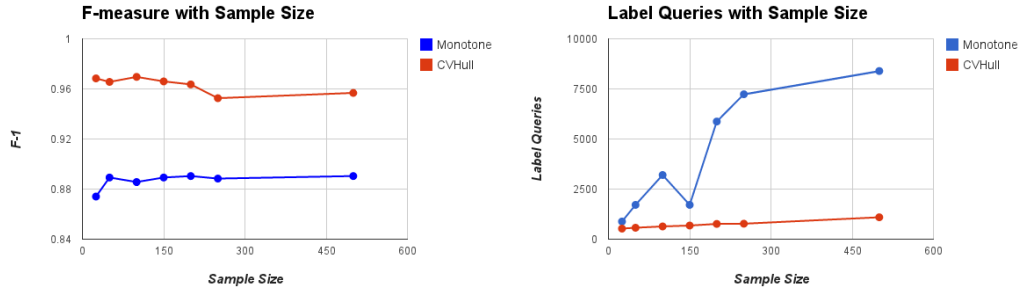
Binary Search Δ: In our implementation of CVHULL we did not explicitly enumerate all possible Λ, or even discretize the space of Λ as we described in Section 3.1 to obtain our theoretical guarantees. We simply used a fixed threshold $\Delta = 0.001$ for the binary search, i.e., if the two slopes in our interval during binary search were within Δ of each other, we stopped the binary search.

We now study how this threshold impacts the peformance of our algorithm. We fix the precision threshold to be 0.85, as before. In Figure 9, we plot the F-1 score and label complexity of CVHULL as a function of Δ, with Δ varying between 0.001 and 0.5, for the DBLP-ACM dataset and the Person dataset. As can be seen in the figure, the F-1 score is not sensitive to Δ, as long as $\Delta < 0.3$, for both datasets. The same holds true for the label complexity. Moreover, we find that if the Δ is set large, then the number of



(a) Sample Size vs. F1 (b) Sample Size vs. Queries

Fig. 7: The effect of varying sample size on F-measure and number of label queries on the DBLP-ACM dataset.



(a) Sample Size vs. F1 (b) Sample Size vs. Queries

Fig. 8: The effect of varying sample size on F-measure and number of label queries on the Person dataset.

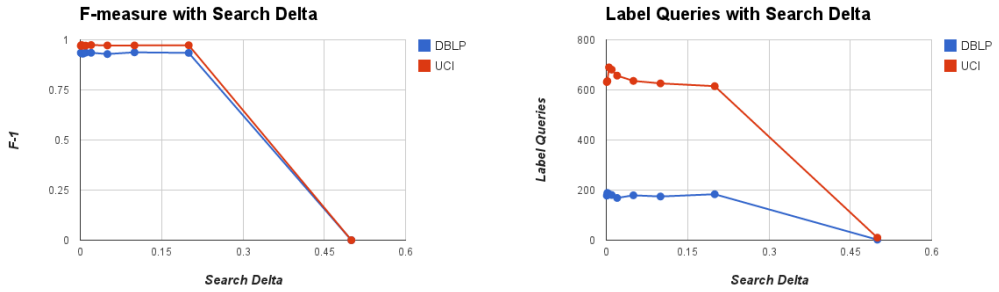
label queries reduces to near zero. This behavior can be used as an indication that we have set the Δ too large; we can then reduce Δ appropriately to get a better classifier.

5.3.2. Deeper Understanding of CVHULL. In this subsection, we focus on gaining a better understanding of the progress of CVHULL over iterations.

Label Usage: So far, we have gathered only a very sparse understanding of where our algorithm uses label queries: does it take a large number of binary search iterations, with a few label queries per iteration, or does it take few binary search iterations, with many label queries per iteration? How does the number of label queries vary with the binary search iteration number?

To get a better understanding of where our algorithm utilizes label queries, we plot the number of label queries as a function of the iteration number in Figure 10, and the average number of queries required as a function of the precision threshold in Figure 11 for the DBLP-ACM dataset.

Figure 10 shows that the number of label queries is large in the first few binary search iterations and then rapidly reduces to zero. The search quickly converges in 7 iterations. Figure 11 shows that the average number of queries required increases with the increase in precision threshold. This behavior is in line with our intuition that



(a) Δ vs. F1 (b) Δ vs. Queries

Fig. 9: The effect of varying the binary search Δ on F-measure and number of label queries on the DBLP and Person datasets.

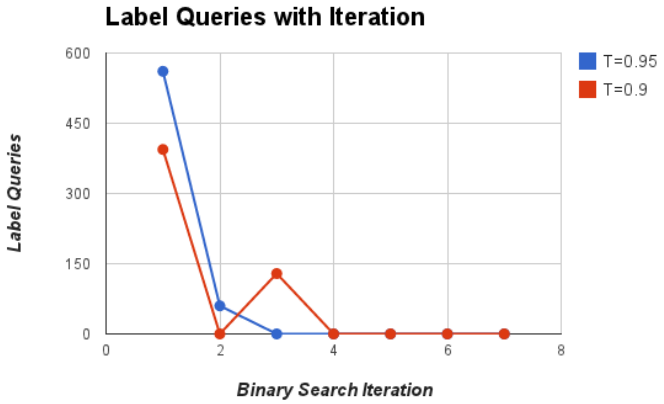


Fig. 10: The number of label queries as a function of the binary search iteration on the DBLP-ACM dataset.

a better classifier is harder to obtain, i.e., requires more labeled examples. However, the number of label queries scales only linearly in precision.

Progress of Algorithm: In Figure 12, we plot the F-1 score and precision as a function of the iteration number of binary search, while fixing the precision thresholds at 0.9 and 0.95, for DBLP-ACM dataset.

As can be seen in the figure, the algorithm very quickly converges (within two iterations) to the best F-1 score, after which the F-1 score and precision (and therefore recall) stay relatively stable. This result shows that we may in fact be able to terminate our algorithm early and get a F-1 score close to optimal. Also note that the final classifier in both cases satisfies the precision threshold.

5.3.3. Synthetic Data. We now illustrate some properties of our algorithm on synthetic data. Synthetic data allows us to have better control over the dataset, and check if our algorithm is able to find the optimal classifier for that dataset.

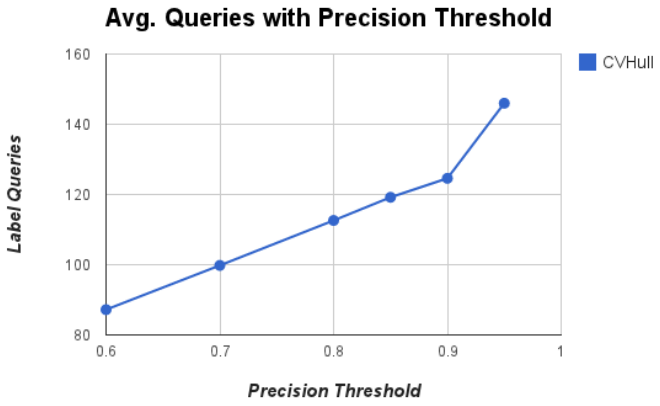
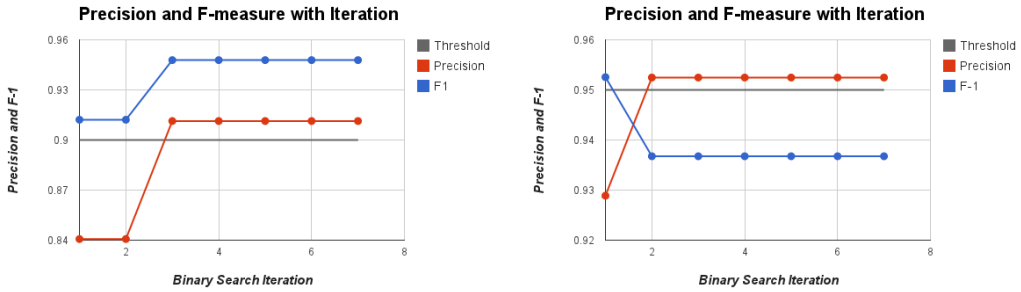


Fig. 11: The number of label queries as a function of the precision threshold on the DBLP-ACM dataset.

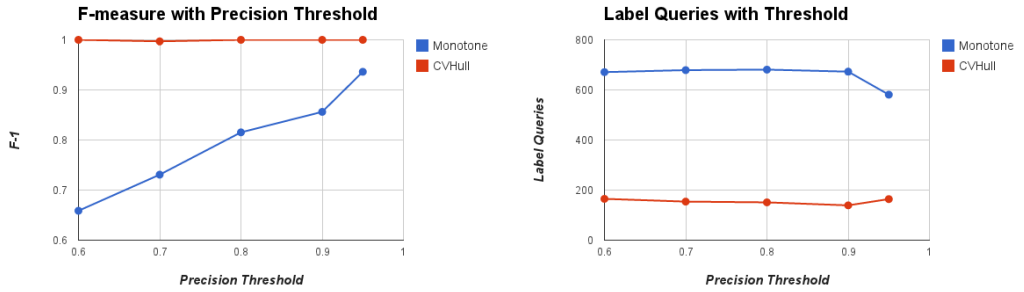


(a) Threshold=0.9 (b) Threshold=0.95

Fig. 12: The precision and F-1 as a function of the binary search iteration on the DBLP-ACM dataset at thresholds $\tau \in \{0.9, 0.95\}$.

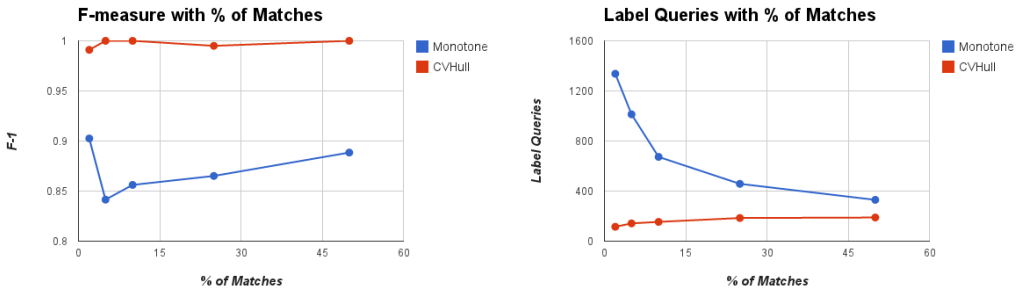
Closeness to Optimal: So far, we have experimented with datasets which have large quantities of noise. We would now like to see how our algorithm and MONOTONE perform when given a separable dataset.

We generated the separable dataset of 5 dimensions by fixing random thresholds for each of the dimensions. We then generate a random set of matching points where the points all have individual dimension values higher than the random thresholds for the corresponding dimensions, and a random set of non-matches, where the non-matches all have individual dimension values lesser than the random thresholds for the correspinding dimensions. Thus there are an infinite number of optimal classifiers, each with F-1 score = 1. (Note that we generated the data in this manner so that there are optimal linear as well as threshold classifiers for this dataset — while CVHULL generates linear classifiers, MONOTONE generates a threshold classifier. Thus, in an ideal scenario, both CVHULL and MONOTONE should be able to find a classifier with F-1 score 1.)



(a) Threshold vs F-1 (b) Threshold vs Queries

Fig. 13: The F-1 and label queries as a function of precision threshold on the synthetic dataset with 10% matches.



(a) Match% vs F-1 (b) Match% vs Queries

Fig. 14: The F-1 and label queries as a function of percentage of matches in the synthetic dataset.

The fraction of matches in this dataset was fixed at 10%. We plot the label complexity and F-1 for both CVHULL and MONOTONE as a function of the precision threshold (Figure 13). We find that our algorithm has much better F-1, in fact nearly close to 1 for each setting, while MONOTONE is much worse. Moreover, CVHULL uses only 1/3 the labels that MONOTONE uses, and the number of labels required does not change as the precision threshold increases (presumably because the same classifier is optimal for all precision thresholds, and a similar number of labels are required to find that classifier.)

Imbalance: We now study the impact of the imbalance between matches and non-matches on MONOTONE and CVHULL. Once again, we use a synthetically generated dataset, with points generated as described in the previous experiment. Here, we control the number of match points generated, and the number of non-matches.

We define the match percentage to be the ratio between the number of matches and the total number of points in the dataset (lower the match percentage, the more imbalanced the dataset is.) In Figure 14, we plot label complexity and F-1 as a function of the percentage of matches in the dataset, where the percentage varies from 2% to 50%.

As can be seen in the figure, our algorithm requires almost the same number of queries at various match percentages and achieves almost perfect F-1 scores across all datasets. Moreover, the most gain from our algorithm relative to MONOTONE is achieved when the data is heavily imbalanced, which is the common scenario in entity matching.

6. CONCLUSION

In this paper, we proposed an active learning algorithm for the entity matching problem. The algorithm tries to learn a classifier with maximum recall under a constraint that its precision should be greater than a given threshold. The algorithm uses any of the existing active learning technique for minimizing 01-loss as a black-box. We showed that the algorithm outputs a classifier having recall close to the optimal, and has good label and computation complexity. We also compared the algorithm against the state-of-the-art active learning algorithm for entity matching, and show that we outperform it in terms of metrics such as F1 of the trained classifier, number of labeled examples required, and computation time on several real-world datasets. Moreover, our algorithm is robust to changes in parameters or datasets compared to other algorithms.

REFERENCES

ARASU, A., GÖTZ, M., AND KAUSHIK, R. 2010. On active learning of record matching packages. In *SIGMOD Conference*. 783–794.

BALCAN, M.-F., BEYGELZIMER, A., AND LANGFORD, J. 2009. Agnostic active learning. *J. Comput. Syst. Sci. 75,* 1, 78–89.

BALCAN, M.-F., BRODER, A. Z., AND ZHANG, T. 2007. Margin based active learning. In *COLT*. 35–50.

BEYGELZIMER, A., DASGUPTA, S., AND LANGFORD, J. 2009. Importance weighted active learning. In *ICML*. 7.

BEYGELZIMER, A., HSU, D., LANGFORD, J., AND ZHANG, T. 2010. Agnostic active learning without constraints. In *NIPS*. 199–207.

BILENKO, M. AND MOONEY, R. J. 2003a. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '03. ACM, New York, NY, USA, 39–48.

BILENKO, M. AND MOONEY, R. J. 2003b. On evaluation and training-set construction for duplicate detection. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage and Object Consolidation, Washington DC*. 7–12.

CHAUDHURI, S., CHEN, B.-C., GANTI, V., AND KAUSHIK, R. 2007. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd international conference on Very large data bases*. VLDB '07. VLDB Endowment, 327–338.

CHRISTEN, P. 2005. Probabilistic data generation for deduplication and data linkage. In *IDEAL*. 109–116.

COHN, D., ATLAS, L., AND LADNER, R. 1994. Improving Generalization with Active Learning. *Mach. Learn. 15,* 2, 201–221.

DASGUPTA, S. AND HSU, D. 2008. Hierarchical sampling for active learning. In *ICML*. 208–215.

DASGUPTA, S., HSU, D., AND MONTELEONI, C. 2007. A general agnostic active learning algorithm. In *NIPS*.

DASGUPTA, S. AND LANGFORD, J. 2009. Tutorial summary: Active learning. In *ICML*. 178.

ELMAGARMID, A. K., IPEIROTIS, P. G., AND VERYKIOS, V. S. 2007. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng. 19*, 1–16.

FRANK, A. AND ASUNCION, A. 2010. UCI machine learning repository.

HANNEKE, S. 2007. A bound on the label complexity of agnostic active learning. In *ICML*. 353–360.

HANNEKE, S. 2009. Adaptive rates of convergence in active learning. In *COLT*.

HUANG, S.-J., JIN, R., AND ZHOU, Z.-H. 2010. Active learning by querying informative and representative examples. In *NIPS*. 892–900.

KARAMPATZIAKIS, N. AND LANGFORD, J. 2011. Online importance weight aware updates. In *UAI*. 392–399.

KÖPCKE, H. AND RAHM, E. 2008. Training selection for tuning entity matching. In *QDB/MUD*. 3–12.

KÖPCKE, H. AND RAHM, E. 2010. Frameworks for entity matching: A comparison. *Data Knowl. Eng. 69*, 197–210.

MCCALLUM, A., NIGAM, K., AND UNGAR, L. H. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '00. ACM, New York, NY, USA, 169–178.

MINEIRO, P. 2003. Cost-Sensitive Binary Classification and Active Learning.

PREPARATA, F. P. AND SHAMOS, M. I. 1985. *Computational geometry: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA.

RENDLE, S. AND SCHMIDT-THIEME, L. 2008. Active learning of equivalence relations by minimizing the expected loss using constraint inference. In *ICDM*. 1001–1006.

SARAWAGI, S. AND BHAMIDIPATY, A. 2002. Interactive deduplication using active learning. In *KDD*. 269–278.

SETTLES, B. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

TEJADA, S., KNOBLOCK, C. A., AND MINTON, S. 2001. Learning object identification rules for information integration. *Inf. Syst. 26*, 607–633.

WANG, W. AND ZHOU, Z.-H. 2010. Multi-view active learning in the non-realizable case. In *NIPS*. 2388–2396.

WHANG, S. E., MENESTRINA, D., KOUTRIKA, G., THEOBALD, M., AND GARCIA-MOLINA, H. 2009. Entity resolution with iterative blocking. In *Proceedings of the 35th SIGMOD international conference on Management of data*. SIGMOD '09. ACM, New York, NY, USA, 219–232.

WINKLER, W. E. 1999. The state of record linkage and current research problems. Tech. rep., Statistical Research Division, U.S. Census Bureau.

ZADROZNY, B., LANGFORD, J., AND ABE, N. 2012. Cost-Sensitive Learning by Cost-Proportionate Example Weighting.