Active sampling for entity matching

Article · January 2012

DOI: 10.1145/2339530.2339707

4 authors, including:

Vibhor Rastogi
Twitter
25 PUBLICATIONS **931** CITATIONS

SEE PROFILE

Active Sampling for Entity Matching

Kedar Bellare
Yahoo! Research
kedar@yahoo-inc.com

Suresh Iyengar
Yahoo! Research
supartha@yahoo-inc.com

Aditya Parameswaran
Stanford University
adityagp@cs.stanford.edu

Vibhor Rastogi
Yahoo! Research
rvibhor@yahoo-inc.com

ABSTRACT

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 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 work [1] 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 under precision constraint with provably sub-linear label complexity (under certain distributional assumptions). Our algorithm uses as a black-box any active learning approach 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.

1. INTRODUCTION

Entity Matching (EM) is the problem of determining if two entities in a data set 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 [10]).

Machine learning approaches [24, 6, 5] 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 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 [9, 13, 2, 12, 4, 3] are thus used that attempt 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 [23, 16] 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 work [1] considers precision and recall[1] of the classifier in the objective function. 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.

[1] gives 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 the classifiers in the high-dimensional parameter space. Due to this high-dimensional binary search, the algorithm has high label complexity (i.e., many labeled example requested) and 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

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

black-box any active learning approach that minimizes 0-1 loss. We first use a Langrange 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 Langrange multipier, the linear objective can be optimized using the given black-box. Then we search for the right value of the Langrange multiplier. For this we embed all classifiers in a two dimensional space, and perform a search along the convex hull [18] 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, [1] has exponential (in the number of dimensions) worst-case computational complexity.

While the classifier output by our algorithm need not be optimal in terms of its recall, we show that it is pareto-optimal (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 a pareto-optimal, and close to the true optimal in terms of recall.
- We use the IWAL active learning algorithm [4] 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 than 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 data sets and compare it with the technique in [1]. 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 $n = |C|$ 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*

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

The RECALL problem is difficult to solve because recall(h) and precison(h) are complicated 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*

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

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.

[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.

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 [7]. 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 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 [7], several approaches have been proposed to handle non-separable datasets. Most recently, IWAL [4] 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 in interest. The algorithm guarantees the following properties.

THEOREM 2.2 (IWAL [4]). *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 [12], is a parameter depending on the source distribution and the hypothesis class, but independent of the number of points* n*.*

Under certain distributional assumption [4], 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, $fn(h)/n$. 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 $\epsilon = \tau/(1-\tau)$, 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 RECALLWEIGHTED.

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 $\alpha = 0$, 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 RECALLWEIGHTED problem by repeatedly solving the 01-LOSSWEIGHTED problem. Note that the objective of both problems is the same, but RECALLWEIGHTED has an additional 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.

$$\begin{aligned} \alpha(h) &= \frac{-(\alpha fn(h) + (1-\alpha) fp(h))}{n} \\ \beta(h) &= \frac{\epsilon \cdot tp(h) - fp(h)}{n} \end{aligned}$$

Thus, the RECALLWEIGHTED problem is equivalent to finding a classifier h that has the highest $\alpha(h)$ under the constraint that $\beta(h) \geq 0$. The 01-LOSSWEIGHTED problem can be shown to be equivalent to finding a classifier h that maximizes $\alpha(h) + \lambda\beta(h)$ for a given λ.

Let $P = \{(\alpha(h), \beta(h)) : h \in H\}$ 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 $(\alpha(h), \beta(h))$ lies on an edge of C. Any classifier h lying on C is pareto-optimal: no other classifier h' exists that has both $-\alpha(h')$ (i.e. objective) as well as $\beta(h')$ (i.e. precision) better than h. We will show that CONVEXHULL algorithm returns classifiers on the convex hull. (Hence the name.)

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 $\alpha(h)$ and $\beta(h)$ are functions of $fn(h)$, $fp(h)$, and $tp(h)$, each of which can vary from 0 to n.

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. Any slope is a ratio of two differences, each with at most $O(n)$ values, and hence both S and Λ have at most $O(n^2)$ values.

The CONVEXHULL Algorithm. For the algorithm, we assume a black-box B for the 01-LOSSWEIGHTED problem that takes a λ and returns a h maximizing $\alpha(h) + \lambda\beta(h)$. 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 $\alpha(h) + \lambda\beta(h)$ using the black-box B. Depending on whether $\beta(h) \geq 0$ or not, the intervals are

Input: A set of n pairs X,
Black-box $B(\lambda)$ returning h with max $\alpha(h) + \lambda\beta(h)$
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 $\beta(h) \geq 0$ set $max = mid$, else set $min = mid$
end while
return h

Algorithm 1: ConvexHull 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 $B_h(\bar{x}, \bar{y})$

Algorithm 2: RejectionSampling Algorithm

updated to ensure that we always have at least one extreme point that satisfies feasibility requirement $\beta(h) \geq 0$. Finally the algorithm terminates with a feasible solution, i.e. a classifier h with $\beta(h) \geq 0$. We can show following additional properties of the algorithm.

Theorem 3.1 (Complexity of ConvexHull). *The* ConvexHull *algorithm terminates with a feasible classifier (i.e. classifier satisfying the precision constraint) with label and computation complexity at most* $2 \log n$ *times that of black-box* B *used in the algorithm.*

Proof. Since size of Λ is bounded by $O(n^2)$, the the binary search procedure finishes in at most $2 \log n$ 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 $\beta(h) \geq 0$, i.e satisfies the precision constraint. This is always true, for a linear classifier labeling all points as negative. □

On the optimality of the solution: 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 $\beta = \alpha$ 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 $\alpha + \lambda\beta$ 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 $\alpha + \lambda\beta$. 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. $\beta(h) \geq 0$. If $-1/\lambda_0$ is a slope between 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. □

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 [25, 17]. 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 can not decide its rejection probability. To overcome this we query B to check whether 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 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. While doing this it gives the black-box 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 $\frac{\alpha fn(h)+(1-\alpha)fp(h)}{n}$, 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-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. 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 [4]. 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 $\alpha(h) + \lambda\beta(h)$, 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 [4]: 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 [4]. 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 [4], 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 $\alpha' = \frac{1-1/\log n+\epsilon\lambda}{1+(1+\epsilon)\lambda}$, 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.

Active learning for entity matching. The previous work most similar to ours is [1], which also provides an active learning algorithm to maximize recall under a constraint that precision should be greater than a specified threshold. This objective function is especially suitable for the class imbalanced entity matching problem. However, their 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 along each dimension in the high-dimensional parameter space. Due to this high-dimensional search, the algorithm has high label and computational complexity, and in the worst-case requires the labels of all n input pairs, as well as exponential in dimensions computational complexity. 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.

[19, 22] also study the problem of using active learning for deduplication. However, the approaches proposed does not provide any guarantees in terms of precision or recall, which is undesirable for highly imbalanced datasets. [21] considers the problem of classification when the people providing the labels may be noisy and then studies the tradeoff between requesting the same label again versus requesting new labels. This work is not applicable to our setting since we use in-house experts to provide labels for each pair requested.

Active learning for general classification. Almost all related work on active learning for the general binary classification problem focuses on 0-1 loss minimization. One of the first active learning algorithms was given in [7]. The algorithm assumes separability of the datasets, i.e. it assumes that a classifier exists having 0 empirical 0-1 loss. The paper showed that a good classifier can be learnt for separable problems using only $O(\log n)$ labeled examples. The algorithm was based on the idea of selective sampling: each example point is queried with a probability, computed based on the point, and also on previously labeled examples.

Since [7], several recent approaches have been proposed to handle non-separable datasets. Most recently, IWAL [4] 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 in interest.

To our knowledge, [3] is the only active learning algorithm that minimizes general loss functions in addition to 0-1 loss. However, the algorithm can not 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 [2, 12, 8] also required maintaining a candidate set of hypotheses (called a version space), which is computationally infeasible.

For a good overview of active learning, see [20]. [9] provides a good summary of recent work in the area.

Entity Matching. Many techniques have been proposed for the entity matching problem [10, 15]. The ones most relevant to us are learning-based techniques that train a classifier over labeled pairs of examples. These include naive bayes [24], decision trees [6], SVMs [5]. 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, [14] describes an algorithm that uses a heuristic string similarity function and then samples pairs having varied similarity scores. The algorithm can not 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.

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 [1] on several real-world datasets (Section 5.2).

5.1 Setup

5.1.1 Data Sets

A brief description of the four real-world datasets used in our experiments is shown in Table 1. 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 [14]. 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 [11]. It contains 574913 pairs for which features have been computed and labels have been provided.
- DBLP-ACM [14]: This is a large bibliography record link-

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

Table 1: 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.

age task in which 494437 matching records in dblp and acm have been manually labeled as either duplicates or non-duplicates.

- Scholar-DBLP [14]: This is a dataset of 589326 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.

5.1.2 Algorithms

We implement three algorithms for active learning.

- MONOTONE: This is our implementation of the algorithm described in [1]. The precision evaluations required for the algorithm is done over random samples of size 100. To ensure 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 [4] obtained from the open source Vowpal Wabbit [13]. 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 Results

In Fig. 1, 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 at high precision thresholds are limited, possibly owing to limited choices of classifiers satisfying the precision constraint.

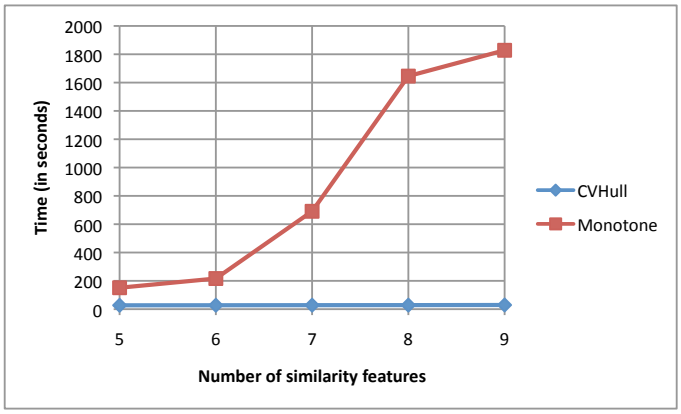
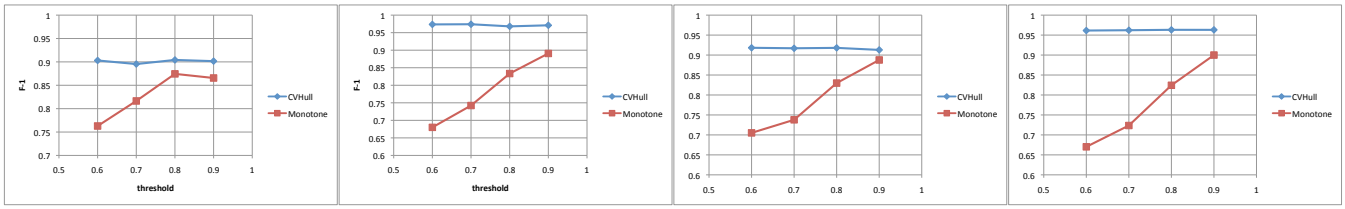


Figure 3: The computational complexity of the two algorithms as the dimension is varied.

In Fig. 2, 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 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. Particularly attractive property of CVHULL is that it learns the classifier in around 500 points for all but one datasets and all threshold values.

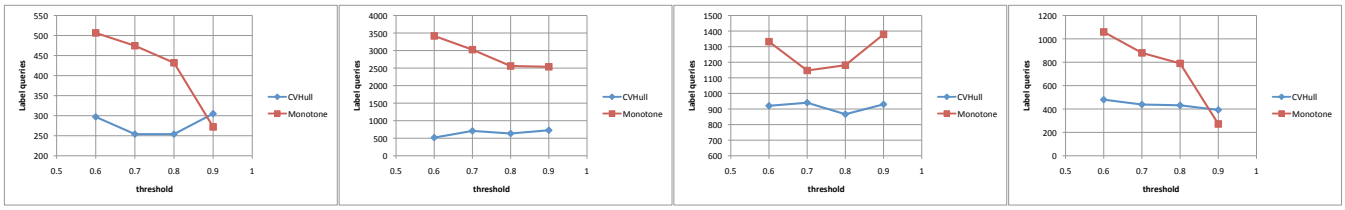
In Fig. 3, we compare the computation time for the two algorithms MONOTONE and CVHULL as the number of dimensions is varied for the Person dataset. The figure clearly demonstrates that MONOTONE has an exponential complexity w.r.t number of dimensions, while CVHULL has an almost a constant dependence.

In Fig. 4, 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



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

Figure 1: The effect of varying the precision threshold on the F-measure for different datasets.



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

Figure 2: The effect of varying the precision threshold on the number of label queries for different datasets.

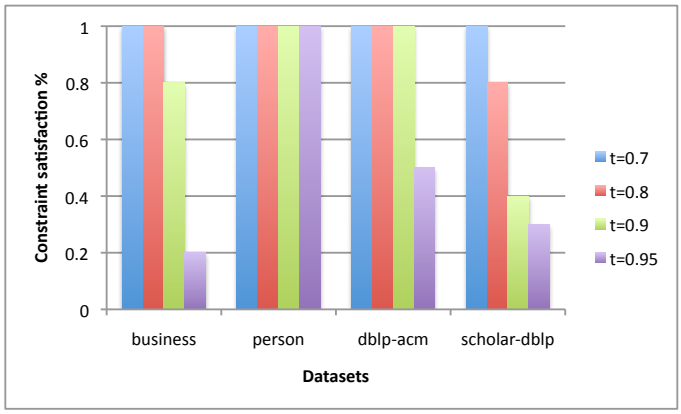


Figure 4: The constraint satisfaction percentage of VW active learning algorithm over 10 random runs for different datasets.

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.

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 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.

7. REFERENCES

[1] Arvind Arasu, Michaela Götz, and Raghav Kaushik. On active learning of record matching packages. In *SIGMOD Conference*, pages 783–794, 2010.

[2] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. *J. Comput. Syst. Sci.*, 75(1):78–89, 2009.

[3] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *ICML*, page 7, 2009.

[4] Alina Beygelzimer, Daniel Hsu, John Langford, and Tong Zhang. Agnostic active learning without constraints. In *NIPS*, pages 199–207, 2010.

[5] Mikhail Bilenko and Raymond J. Mooney. 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, pages 39–48, New York, NY, USA, 2003. ACM.

[6] Surajit Chaudhuri, Bee-Chung Chen, Venkatesh Ganti, and Raghav Kaushik. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 327–338. VLDB Endowment, 2007.

[7] David Cohn, Les Atlas, and Richard Ladner. Improving Generalization with Active Learning. *Mach. Learn.*, 15(2):201–221, May 1994.

[8] Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In *NIPS*, 2007.

[9] Sanjoy Dasgupta and John Langford. Tutorial summary: Active learning. In *ICML*, page 178, 2009.

[10] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 19:1–16, January 2007.

[11] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[12] Steve Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, pages 353–360, 2007.

[13] Nikos Karampatziakis and John Langford. Online importance weight aware updates. In *UAI*, pages 392–399, 2011.

[14] Hanna Köpcke and Erhard Rahm. Training selection for tuning entity matching. In *QDB/MUD*, pages 3–12, 2008.

[15] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69:197–210, February 2010.

[16] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 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, pages 169–178, New York, NY, USA, 2000. ACM.

[17] Paul Mineiro. Cost-Sensitive Binary Classification and Active Learning, 2003.

[18] Franco P. Preparata and Michael I. Shamos. *Computational geometry: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[19] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *KDD*, pages 269–278, 2002.

[20] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[21] V. S. Sheng, F. Provost, and P. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *SIGKDD*, pages 614–622, 2008.

[22] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Inf. Syst.*, 26:607–633, December 2001.

[23] Steven Euijong Whang, David Menestrina, Georgia Koutrika, Martin Theobald, and Hector Garcia-Molina. Entity resolution with iterative blocking. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 219–232, New York, NY, USA, 2009. ACM.

[24] William E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

[25] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-Sensitive Learning by Cost-Proportionate Example Weighting, 2012.