Interactive Deduplication using Active Learning

Sunita Sarawagi
sunita@it.iitb.ac.in
IIT Bombay

Anuradha Bhamidipaty
anu@it.iitb.ac.in
IIT Bombay

Abstract

Deduplication is a key operation in integrating data from multiple sources. The main challenge in this task is designing a function that can resolve when a pair of records refer to the same entity in spite of various data inconsistencies. Most existing systems use hand-coded functions. One way to overcome the tedium of hand-coding is to train a classifier to distinguish between duplicates and non-duplicates. The success of this method critically hinges on being able to provide a *covering and challenging* set of training pairs that bring out the subtlety of the deduplication function. This is non-trivial because it requires manually searching for various data inconsistencies between any two records spread apart in large lists.

We present our design of a learning-based deduplication system that uses a novel method of interactively discovering challenging training pairs using *active learning.* Our experiments on real-life datasets show that active learning significantly reduces the number of instances needed to achieve high accuracy. We investigate various design issues that arise in building a system to provide interactive response, fast convergence, and interpretable output.

1 Introduction

A crucial step in integrating data from multiple sources is detecting and eliminating duplicate records that refer to the same entity. This process is called *deduplication.* Large customer-oriented organizations often merge long lists of names and addresses with partially overlapping sets of customers. Another area where deduplication is necessary is in the construction of web portals which integrate data from various pages possibly created in a distributed manner by millions of people. Examples of such portals are CiteSeer [16] and Cora [18] that integrate citations and paper titles parsed and extracted from several personal and publisher webpages.

The main challenge in this task is finding a function that can resolve when two records refer to the same entity in spite of errors and inconsistencies in the data. We motivate the difficulty of manually designing a good deduplication function with examples from the bibliography domain.

Illustrative Example All citations in this illustration are from the CiteSeer citation database (`http://citeSeer.nj.nec.com/cs`). The entries that we show as duplicates in these examples are the ones that escaped detection by CiteSeer's manually tuned deduplication function [16].

Based on our domain knowledge, we know that two citations are duplicates if their author, title, year, and where-published fields match. However, citations come in a large spectrum of formats making a specific codification of our knowledge difficult.

We show below two example duplicate citations of a book — these are just two of the 14 groups of citations to the book that Citeseer did not collapse as duplicates[1]

- L. Breiman, L. Friedman, and P. Stone, (1984). Classification and Regression. Wadsworth, Belmont, CA.
- Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. Classification and Regression Trees. Wadsworth and Brooks/Cole, 1984.

The example brings out the non-triviality of author match. Individual author names in a multi-author list might be hard to separate, orders of names could be reversed, some names could be missing, misspelt, or abbreviated. Matching of conference names could also get tricky as seen in the duplicates below:

- R. Agrawal, R. Srikant. Fast algorithms for mining association rules in large databases. In VLDB-94, 1994.
- Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

VLDB-94 is not a straightforward abbreviation of the booktitle in the second citation. The year field might also have errors — this is commonplace when one of the citation has "to appear".

Another difficulty is the lack of an explicit structure in many input data sources. Citations often do not come segmented into the author/title/conference fields. Typically an automated program is used for the segmentation [2]. This program cannot always do a perfect job, thereby introducing another kind of inconsistency. A natural approach then is to forget about matching individual entries thus and simply attempt an overall count of the number of common words. In the example below we show two citations that are not duplicates even though they have a large number of common words.

- H. Balakrishnan, S. Seshan, and R. H. Katz., Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks, ACM Wireless Networks, 1(4), December 1995.
- H. Balakrishnan, S. Seshan, E. Amir, R. H. Katz, "Improving TCP/IP Performance over Wireless

[1] Obtained by searching for citations with keyword "breiman" in CiteSeer, `http://citeseer.nj.nec.com/cs?q=breiman&submit=Search+Citations&cs=1`

Networks," Proc. 1st ACM Conf. on Mobile Computing and Networking, November 1995.

In contrast, the two citations below are duplicates in spite of having significantly fewer common words even in the title.

- Johnson–Laird, Philip N. (1983). Mental models. Cambridge, Mass.: Harvard University Press.
- P. N. Johnson-Laird. Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness. Cambridge University Press, 1983.

State of the art Most existing systems for deduplication are based on hand-coded rules that compute the extent of match between individual attributes and combine the match scores by using thresholds and several if-then- else conditions.

One way of reducing the tedium of hand-coding is to relegate the task of finding the deduplication function to a machine learning algorithm [31, 32, 6]. The algorithm takes as input training examples consisting of pairs of duplicates and non-duplicates. A second input is a collection of various kinds of simple, domain-specific matching functions on various attributes, provided by a domain expert. The learning algorithm can then use the examples to automatically find the best way of combining and thresholding the attribute similarity scores.

Limitations of the learning approach The success of the above learning method crucially hinges on being able to provide a *covering and challenging* set of duplicates and non-duplicates that will bring out the subtlety of the deduplication function. Finding such examples is hard because this requires a user to manually do a quadratic search in large records lists where the confusing record pairs might be spread far apart. Surprisingly, what is even harder is finding an interesting set of non-duplicates that are likely to be confused as duplicates in an existing training set. Even though non-duplicates abound, it is hard to find non-duplicates that can effectively counter any simple-minded deduplication function inferred from an existing training set.

Our contribution We designed a learning based deduplication system (ALIAS[2]) that allows automatic construction of the deduplication function by using a novel method of interactively discovering challenging training pairs. Our key insight is to simultaneously build several redundant functions and exploit the disagreement amongst them to discover new kinds inconsistencies amongst duplicates in the dataset. **Active learning** [7, 9] methods also rely on a similar insight for selecting instances for labeling from a large pool of unlabeled instances. Unlike an ordinary learner that trains using a static training set, an active learner actively picks subsets of instances which when labeled will provide the highest information gain to the learner.

With this approach the more difficult task of bringing together the potentially confusing record pairs is automated by the learner. The user has to only perform the easy task of labeling the selected pairs of records as duplicate or not.

We designed an active learning algorithm that can meet our design goals of interactive response, fast convergence, and high accuracy. Finally, our system outputs a deduplication function that is easy to interpret and efficient to evaluate when deployed on large record lists. This required evaluating various non-obvious design tradeoffs that arise when using current active learning methods in a practical setting. Experiments on real-life datasets show that our approach reduces the number of labeled training pairs by two orders of magnitude to reach a certain accuracy. After labeling less than 100 pairs selected interactively by our system, the learnt deduplication function can achieve the peak accuracy which a randomly chosen set of pairs cannot achieve even with 7000 pairs.

[2] *Active Learning led Interactive Alias Suppression*

Outline There are two main parts to the paper. The first part (Section 2) is an overall description of our ALIAS interactive deduplication system. The second part starting from Section 3 covers the main novelty of our system, namely, the mechanism for selecting the informative sets of pairs, the foundations of which lie in Active Learning. Section 4 presents an experimental evaluation of the effectiveness of active learning in easing the deduplication task. Section 5 discusses related work. Finally, conclusions appear in Section 6.

2 Overall architecture

Figure 1 shows the overall design of our ALIAS system for deduplication. There are three primary inputs to the system:

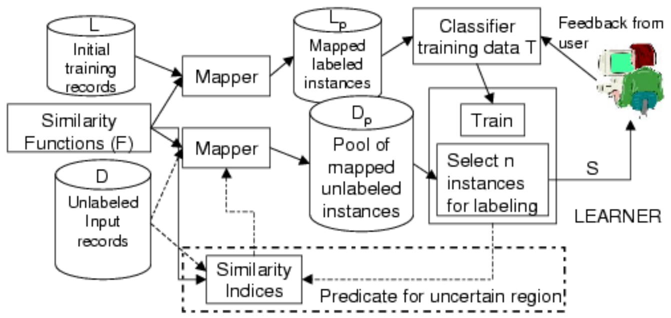
1. Database of records (D) The original set D of records in which duplicates need to be detected. The data has d attributes $a_1, \dots a_d$, each of which could be textual or numeric. The goal of the system is to find the subset of pairs in the cross-product $D \times D$ that can be labeled as duplicates.

2. Initial training pairs (L) An optional small(less than ten) seed L of training records arranged in pairs of duplicates or non-duplicates.

3. Similarity functions ($\mathcal{F}$) A set $\mathcal{F}$ of n_f functions each of which computes a similarity match between two records r_1, r_2 based on any subset of d attributes. Examples of such functions are edit-distance, soundex, abbreviation-match on text fields, and absolute difference for integer fields. Many of the common functions could be inbuilt and added by default based on the data type. However, it is impossible to totally obviate an expert's domain knowledge in designing specific matching functions. These functions can be coded in the native language of the system (C++ in our case) and loaded dynamically. The functions in the set can be highly redundant and unrelated to each other because finally our automated learner wil perform the non-trivial task of finding the right way of combining them to get the final deduplication function.

A rough outline of the main steps is given in Figure 1.

The first step is to map the initial training records in L into a pair format via a mapper module. The mapper module takes as input a pair of records r_1, r_2, computes the n_f similarity functions $\mathcal{F}$ and returns the result as a new record with n_f attributes. For each duplicate pair we assign a class-label of "1" and for all the other pairs in $L \times L$ we assign a class label of "0". At the end of this step we get a mapped training dataset L_p.These L_p instances are used



```
1. Input: $L$, $D$, $\mathcal{F}$.
2. Create pairs $L_p$ from the labeled data $L$ and $\mathcal{F}$.
3. Create pairs $D_p$ from the unlabeled data $D$ and $\mathcal{F}$.
4. Initial training set $T = L_p$
5. Loop until user satisfaction
       • Train classifier $C$ using $T$.
       • Use $C$ to select a set $S$ of $n$ instances from $D_p$ for labeling.
       • If $S$ is empty, exit loop.
       • Collect user feedback on the labels of $S$.
       • Add $S$ to $T$ and remove $S$ from $D_p$.
6. Output classifier $C$
```

Figure 1: Overall design and working of the ALIAS interactive deduplication system.

to initialize the learning component of the system. More details of this component are deferred to Section 2.2.

The next step is to map the unlabeled record list D. The mapper is invoked on each pair of records in $D \times D$ to generate an unlabeled list of mapped records D_p. If the size of D is large the quadratic size of the cross-product could be intolerable. Later in section 2.1 we discuss how to avoid this through proper indexing (the part within dashed boundaries in Figure 1).

We next describe the interactive active learning session on D_p with the user as the tutor. The learner chooses from the set D_p a subset S of n (a user-configurable parameter, typically less than five) instances that it would most benefit from labeling. Details of how it does this are deferred until Section 3. The user is shown the set of instances S along with the current prediction of the learner. The user corrects any mistakes of the learner. The newly labeled instances in S are added to the training dataset L_p and the active learner is retrained. The user can inspect the trained classifier and/or evaluate its performance on a known test dataset. If (s)he is not happy with the learner trained so far, the active learner can select another set of n instances. This process continues in a loop until the user is happy with the learnt model. In each iteration, the user aids the learner by providing new labeled data.

A useful side effect of the user inspecting the model's prediction at each iteration is that, he can discover newer sources of discrepancies and errors in the data and decide to modify his similarity functions or add new ones.

The output of our system is a deduplication function $\mathcal{I}$ that when given a new list of records A can identify which subset of pairs in the cross-product $A \times A$ are duplicates. An example of a sample deduplication function learnt as a decision tree is shown in Section 4.2.

2.1 The indexing component

When the data set D is large, the quadratic size of $D \times D$ could be problematic since we need to be able to provide interactive response to the user during the active learning.

We support three ways of reducing the number of pairs to be generated.

Grouping Sometimes it is possible to find an easy grouping/windowing function that is guaranteed to bring together all duplicates. Examples of such grouping functions are, year of publication for citation entries and the first letter of the last name for address lists. Pairs are formed only within records of a group. Similar windowing ideas have been exploited in [12].

Sampling In the active learning phase, when we are learning a deduplication function, we may not need to work on the entire set of records D. Sampling appears like a natural recourse in such cases. However, simple random sampling will not work here because in most cases the number of duplicates will be few and sampling might further diminish such duplicate pairs. For example if only 10% of the records in D have one duplicate each, then a 5% random sample of the data will contain on an average just $0.10 \times 0.05 \times 0.05 = 0.025\%$ duplicates. We propose an alternative grouped sampling approach that hinges on being able to find a grouping function such as above. In this approach we sample in units of a group instead of individual records.

Indexing Another option we support is to index the fields of D such that predicates on the similarity function of the attribute can be evaluated efficiently. For example, a predicate on a similarity function of the form "fraction of common words between two text attributes ≥ 0.4" can be evaluated efficiently by creating an inverted index on the words of the text attribute. Clearly, this cannot be done easily for all possible similarity functions. Edit distance is an example of such a hard to index similarity function. In most cases, however, it is possible to approximate a similarity function f with another function g that is always less than or equal to f. So, a predicate that retrieves all records r with $f(r) \geq C$ can be transformed to a looser predicate of the form $g(r) \geq C$. This predicate can be evaluated via the index and later filtered for exact match. For example, [23, 11] show how an inverted index on the NGrams of a text field can used to lower bound the edit-distance function. Such ideas have been used earlier in [19, 11, 5]

However, we cannot exploit such similarity functions unless we modify our active learning mechanism which as described in the previous section takes as input the materialized pairs $D_p = D \times D$ and chooses a subset n for labeling. We need to modify it to not require all of $D \times D$ at a time. It is possible to design active learners that can first output *predicates* that characterize the pairs likely to be selected by the learner. The predicates can be evaluated using the indices and only the qualifying pairs will be materialized and passed to the learner for further subseting to the n instances. This is just a rough outline of an approach. In this paper we will not have space to discuss this topic further. We defer further details to a future paper. Our main focus here is to first study the efficacy of active learning for deduplication.

2.2 The learning component

The core of the learning component of our system is a classifier that in addition to the usual task of training a model given examples and predicting labels for unknown instances also needs to support active learning as discussed in Section 3. We need to consider four criteria in choosing a classifier: accuracy, interpretability, indexability, and, training efficiency. We evaluate three popular classification methods, Decision trees [24] (D-tree), naive Bayes [21] (NB) and Support Vector Machines [4] (SVMs), on each of these four criteria next.

Accuracy Our classification problem is particularly challenging because deduplication datasets are often severely skewed in their class distribution — the fraction of duplicate pairs could be less than 1% of the non-duplicates. Another concern for such highly skewed data is choosing an appropriate metric for evaluating classifiers. Accuracy could be a misleading metric in such cases. For example a case with just 1% duplicates, a trivial classifier that labels all pairs as non-duplicates will yield 99% accuracy — the same as another classifier that identifies all duplicates correctly but also misclassifies 1% non-duplicates. We choose two metrics for evaluation, recall and precision. Recall r is the fraction of duplicates correctly classified and precision p is the fraction correct amongst all instances actually labeled duplicate. Ideally, we should evaluate our methods on both recall and precision separately since no single metric can combine them in a universally acceptable single number. However, we need a single number for ease of plotting. We use a well-known measure from the information retrieval community called the F**-measure** that is the harmonic mean of the precision and recall values, i.e.,

$$F = \frac{1}{0.5(\frac{1}{p} + \frac{1}{r})} = \frac{2pr}{p+r}. \quad (1)$$

Thus, for the first case, the F value is 0 because recall is 0. For the second case $r = 1$ and p=0.5. Thus, $F = 0.667$. Another measure that we considered was weighted accuracy but we found that weighted accuracy can yield very high values of accuracy even when the precision is extremely poor. For example, with 200 duplicates and 5000 non-duplicates and the classifier confusion matrix is $\begin{bmatrix} 180 & 20 \\ 500 & 4500 \end{bmatrix}$, weighted accuracy is 90% whereas precision is just 28%. F measure is 42%.

We report both recall and precision values in addition to the F measure for critical points.

Interpretability A second important concern is choosing classifiers whose final deduplication rule is easy for a human being to understand, interpret and tune. This is necessary so that a domain expert can cover the limitations of the training data and tune the function where needed. Decision trees are most suited on this criteria.

Indexability Once the model is trained, it will be deployed for finding duplicates in large lists of records. In such cases, we cannot afford to generate the quadratic cartesian product of mapped pairs. Instead as discussed earlier in Section 2.1 we wish to analyze the deduplication function learnt by the classifier and construct appropriate similarity indices or grouping functions. Therefore, we need a classifier like a decision tree whose predicates are simple conjuncts and disjuncts on individual similarity functions instead of classifiers like SVMs and naive Bayes that combine the similarity functions in more complicated ways. Sometimes, it might be possible to post-process a classifier, like naive Bayes to extract indexable predicates as shown in [5].

Efficient training Finally, we want a method that is fast to train because for the interactive active learning phase, during each iteration we would need to retrain a classifier with a larger training dataset. Fortunately, the training data size is not too large in each case, so this criteria is easily met by all three of the above classifiers.

3 Active learning

An active learner starts with a limited labeled and a large unlabeled pool of instances. The labeled set forms the training data for an initial preliminary classifier. The goal is to seek out from the unlabeled pool those instances which when labeled will help strengthen the classifier at the fastest possible rate. What criteria should we use for picking such instances? The initial classifier will be sure about its predictions on some unlabeled instances but *unsure* on most others. The unsure instances are those that fall in the classifier's confusion region. This confusion region is large when the training data is small. The classifier can perhaps reduce its confusion by seeking predictions on these uncertain instances. This intuition forms the basis for one major criteria of active learning, namely, selecting instances about which the classifier(s) built on the current training set is most *uncertain*. We give an example to show how selecting instances based on uncertainty can help reduce a classifier's confusion.

Example: Consider a very simple learner for separating points from two different classes: positive (P) and negative (N) on a straight line as shown in Figure 2. Assume that the set of points are separable by a single point on the line. The initial training set consists of one positive point b (star) and one negative point r (circle) picked randomly from the line. The rest of the points(squares) are unlabeled. The confusion region is the region between r and b. Any unlabeled point x to the left of r and to the right of b will have no effect in reducing this confusion. Hence they will not be selected for active learning. The points in between r and b are the ones about which the classifier is uncertain to varying degree. For any point x in this region, assume that the probability that it is negative is inversely proportional to its distance from r. For simplicity, assume r has a coordinate of 0 and b has a coordinate of 1. Thus, if x has a coordinate of d, the probability that its class is negative (N) is $\Pr(N|x) = 1 - d$ and $\Pr(P|x) = d$. If x were negative, the size of confusion margin would reduce by d, if it were positive the size would decrease by $(1 - d)$. Hence, the expected reduction in the size of the confusion region on adding x to the training set is $\Pr(N|x)d + \Pr(P|x)(1-d) = (1-d)d + d(1-d) = 2d(1-d)$. This achieves the maximum value when $d = 0.5$, that is, the point about whose prediction the classifier has the maximum uncertainty. By including m in the training set, the size of the uncertain region will reduce by half no matter what its label. Any other point say s that is close to the negative boundary but far from the positive boundary could reduce the confusion more if its true label is found to be positive

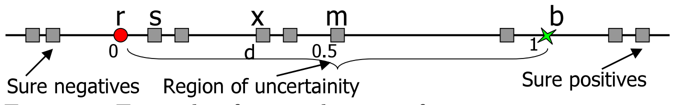


Figure 2: Example of active learning for separating points on a line.

but the probability of that is small given the current training data. Thus, the *expected* reduction in confusion is less for this point than for m — the most uncertain instance.

In the example above, we noticed that the instance about which the learner was most unsure was also the instance for which the expected reduction in confusion was the largest. Instances whose prediction the learner can already make with strong confidence will likely not have much effect on the learner. Theoretical justification for approximating expected reduction in confusion (formally, version space) with prediction uncertainty appear in [30, 1, 9].

The example above was for a simple case where the two classes are completely separable by the classifier. Real-life data is noisy and when picking instances based on uncertainty we need to make sure that we are not picking erroneous or outlying instances. We are more likely to gain from instances that are representative of a large number of unlabeled instances, than an extreme outlying instance. To ensure this, a second criteria that becomes important is the representativeness of an instance.

In the rest of the section we discuss how to quantify the notions of uncertainty (Section 3.1) and representativeness of an instance (Section 3.2). The techniques discussed are a distillation of the various methods of active learning proposed recently [7, 20, 30, 9, 1, 17, 14] along with an explanation of our particular design rationale. We will accompany each design decision with justifications of our chosen approach using experiments on two real-life datasets: a citation database and an address database. More details of the experimental setup can be found in Section 4.

3.1 Uncertainty score of an instance

There are two major ways of evaluating the uncertainty of the prediction on a instance.

An intuitive method for measuring uncertainty for separator based classifiers like SVMs and regression is to make it inversely proportional to the distance of the instance from the separator [30, 27]. Similarly for bayesian classifiers, the posterior probabilities of classes can be used as an estimate of certainty [20]. For decision trees, typically uncertainty is derived from the error of the leaf into which the instance falls [33]. However, in our experiments we found that the uncertainty scores of decision trees and naive Bayes were not satisfactory in returning informative instances.

A classifier independent way of deriving the uncertainty of an instance is by measuring the disagreement amongst the predictions it gets from a committee of N classifiers. The committee is built so that the N member classifiers are slightly different from each other, yet they all have similar accuracy on the training data. The different members thus provide redundant ways of classification. A sure duplicate or non-duplicate would get the same prediction from all members. The uncertain ones will get different labels and by adding them in the training set the disagreement amongst the members will be lowered in the next iteration. Uncertainty of predictions of a committee can be quantified in various ways. We used entropy on the fraction of committee members that predicted each of the two classes.

3.1.1 Methods of creating committees

We next present three different ways of creating committees.

Randomizing model parameters A common method of creating committees is by making small perturbations on the parameters of the model trained through the given training data [28, 1]. The perturbations are made by sampling from a distribution that the particular training parameter is expected to follow. Some of the previous approaches show how to perturb the parameters of a Naive Bayes classifier[20] and Hidden Markov Model [1].

We propose a mechanism for randomizing decision tree classifiers. During tree construction, when selecting an attribute for splitting on next, instead of deterministically choosing the attribute with the highest information gain, we *randomly* pick one with information gain within close range of the best. Secondly, when picking the threshold on which to split a continuous attribute, instead of picking the midpoint of the range within which the information gain remains unchanged, we pick a point uniformly randomly from anywhere in the range.

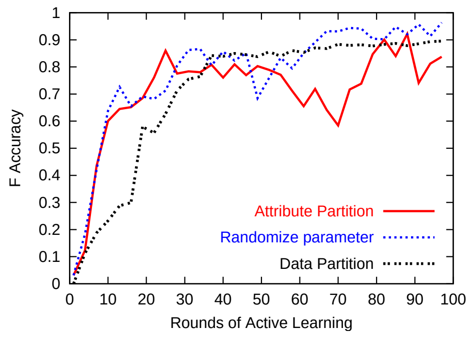
Partitioning training data A second model-independent method of creating committees is by partitioning the training dataset in various ways, including disjoint and overlapping partitioning. Disjoint partitioning did not work well in our experiments due to the limitation of training data in the early phase of active learning. We therefore did N-fold overlapping partitioning where (for a committee of size N) we first partition a training dataset D into N disjoint sets $D_1, D_2 \dots D_N$. Then, train the i-th committee with the dataset $D - D_i$. This way each member gets trained on $(1 - \frac{1}{N})$th fraction of the data.

Attribute Partition When the training data is sparse, but the number of attributes is surplus, another method of constructing a committee is by partitioning the attribute set in various ways. Like for data partitioning, one approach is disjoint partitioning where *all* attributes used in constructing a model are removed from the attribute set before constructing the next model. We stop constructing models once all attributes exhaust or the accuracy on the training data reduces drastically. Not surprisingly, this method did not work well. We therefore used a second approach that removed only the topmost attribute from each earlier model. This is applicable only for decision trees.

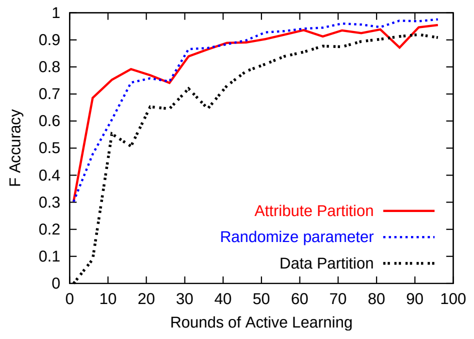
3.1.2 Comparing the three different methods of creating committee

In Figure 3 we compare the three different methods of creating committee members for the two datasets and settings described in Section 4. The x-axis is the different rounds of active learning and since we select one instance per round this is also equal to the number of training instances, the y axis is F accuracy of the current classifier on the total unlabeled data D_p.

We find that for both the datasets, the randomized parameter method performs the best overall, followed by attribute partitioning. Data partitioning is relatively worse



(a) Bibliography data



(b) Address data

Figure 3: Comparing the three different methods of creating committees

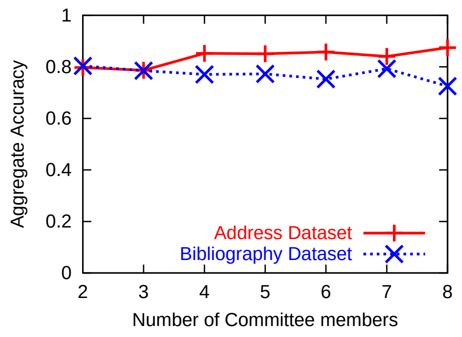


Figure 4: Change in aggregate accuracy with varying number of committee members

in the initial stages when the available training data for partitioning is small. On the contrary, attribute partitioning gets worse in the later stages because as the training set gets larger and more complex, the redundancy in the available attributes diminishes. In such cases, removing the topmost split attributes, makes the subsequent trees significantly weaker.

Number of committee members (N) The committee size N is taken as an input argument. We study its effect on active learning. In Figure 4 we plot committee size versus the area under the accuracy curve formed by active learning with that committee size. The area is measured by summing up the accuracy values for each round. Interestingly, we find that the performance of the classifier is not too sensitive to the number of committee members. This implies that the committee size can be kept to be fairly small (less than five) without hurting accuracy.

3.2 The representativeness of an instance

Real-life data is often noisy. The *most* uncertain instance could often be an outlier. An uncertain instance that is representative of a larger number of unlabeled instances is likely to have a greater impact on the classifier, than an outlying instance. Hence, another important factor is how representative an instance is of the underlying data distribution.

The main challenge in incorporating the representativeness factor is figuring out how to combine it with the uncertainty factor. Two different methods have been proposed for this.

The first approach explicitly measures the representativeness of an instance by estimating the density of points around it after clustering the unlabeled instances [20]. The instances are then scored using a weighted sum of its density and uncertainty value and the n highest scoring instances selected. This method requires us to tune several parameters: distance function for clustering, number of clusters and the weights to tradeoff uncertainty with representativeness.

A second more common approach relies on *sampling* to preserve the underlying data distribution [9, 1, 17]. First, each candidate unlabeled instance is weighted by its uncertainty value. Then the n instances for active learning are selected from this using weighted sampling. We chose and experimented with different variations of this approach starting with no-sampling to full-sampling. In no-sampling we simply pick the n highest uncertainty instances. In full-sampling we do weighted sampling on the entire unlabeled set. An intermediate approach is to first pick the top kn ($k \geq 1$) instances based on uncertainty and then do weighted random sampling on them to select n out of these kn instances. $k = 1$ then corresponds to no sampling, $kn =$ total data size corresponds to the full sampling. We had a default k of 5.

In Figure 5 we show the accuracy change with active learning for the above three sampling schemes with two different base classifiers: decision trees (D-tree), a discriminative classifier and naive Bayes (NB), a generative classifier. For D-trees we find all three sampling schemes to be comparable. We notice the real benefit of accounting for representativeness for naive Bayes classifier. This is expected because for generative classifiers maintaining the input data distribution is much more important than for discriminative classifiers. Theoretical analysis of this phenomenon can be found in [34].

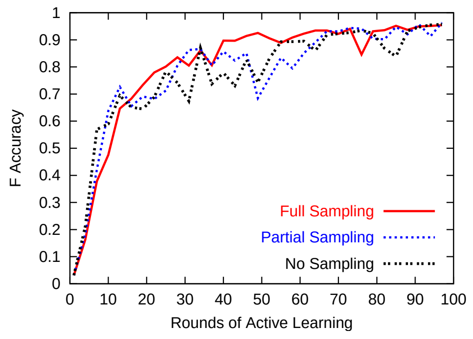
The final algorithm used by our system for picking the n instances for labeling is given in Figure 6.

4 Experimental evaluation

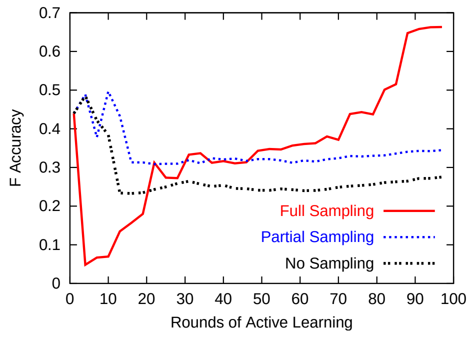
We now present overall evaluation figures for our chosen active learning approach.

Our experiments were on the following two datasets:

The Bibliography dataset consists of citation entries obtained from CiteSeer by searching on the lastnames of the 100 most frequently referred authors. The data consisted of 254 citations, 54 of which were found duplicates (after careful manual searching). In the pair format, this led to $\frac{254\times253}{2} = 32131$ instances of which only 169 were duplicates — that is, only 0.5% of the instances were of the positive



(a) Bibliography data: Decision tree



(b) Bibliography data: Naive Bayes

Figure 5: Comparing different sampling schemes for incorporating representative instances

```
1. Input: $L_p$: current training data, $N$ number of
   committees, $D_p$ unlabeled instances
2. Train $N$ classifiers $C_1, C_2, \ldots C_N$ on $L_p$ by
   randomizing the choice of the parameters for all
   but the first classifier.
3. For each unlabeled instance $x$ in $D_p$,
   (a) Find prediction $y_1 \ldots y_N$ from the $N$ members.
   (b) Compute uncertainty $U(x)$ as the entropy of
       the above $N$ predictions.
4. Return $n$ instances by (weighted) sampling on the
   instances with the weight as $U(x)$.
```

Figure 6: Algorithm used by active learning for selecting n instances for labeling

class. The raw data had no underlying structure. We segmented the text record into five fields namely, author, title, year, page number and rest using CiteSeer's scripts.

The Address dataset consists of names and addresses of customers with the local telephone company of the city of Pune in India. The data had ten attributes: "lastname, firstname, middlename, Address1, $\cdots$ Address6 and Pin". The six address fields did not follow any meaningful breakup of the address. We had 300 records, 98 of which were found duplicates (again by manual search). In the pair format, this led to $\frac{300\times299}{2} = 44850$ instances of which 105 were duplicates — a skewness of 0.25%.

Similarity functions We designed 20 similarity functions on each of the two datasets. For each text attribute, we had three functions: NGrams match (with ngrams of size 3), fraction of overlapping words, and, approximate edit distance as described in [31, 32]. For integer fields like year and page, we had a special number match that tolerated shift by 1. For attributes that are likely to get wrongly segmented as a neighboring field, we created new text fields by concatenating the neighboring fields and defined on them the text matching functions. A special function was also designed for null-matches on each attribute. This is to distinguish cases where two attributes match because they are both nulls, or two attributes mismatch because one of them was null.

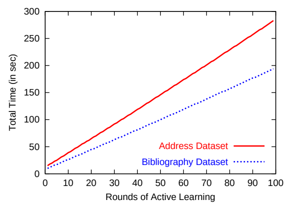


Figure 7: Running time.

Classification methods We used the following three classification methods: C4.5 decision tree classifier, $\mathcal{MLC}$++'s naive Bayes classifier [15], and SVMTorch Support Vector Machine classifier (SVM) [8]. Our experiments were performed on a three processor Pentium III server running Linux redhat 7.0 with 512 MB of RAM.

All our experiments were obtained by averaging over ten runs with different seeds of a random number generator that gets deployed in different stages of our algorithm as discussed in Section 3.

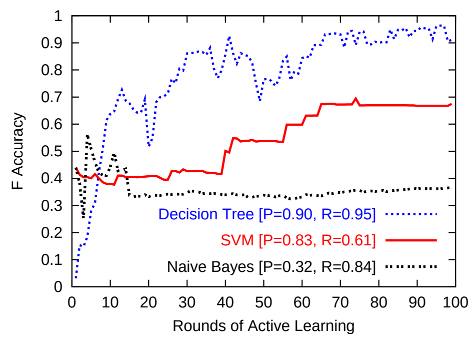
Defaults The defaults for the different parameters of our algorithm were set to the best option found in the experiments of the previous section. The default classification method was decision trees. The number of committees was set to 5. The committees were created using parameter randomization and instances were selected using partial-sampling. The initial training set consisted of one exact duplicate and one clear non-duplicate. In each round of active learning one instance was selected for labeling, i.e., $n = 1$. The accuracy of the classifier at each round of active learning is measured on the entire unlabeled dataset.

4.1 Running time

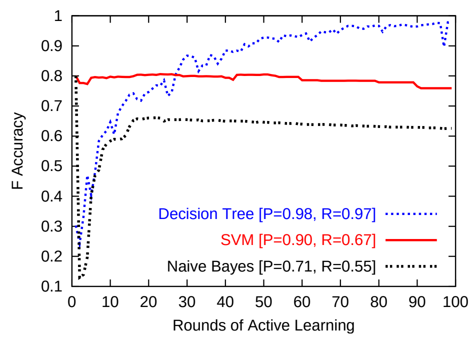
In Figure 7 we plot total running time with increasing rounds of active learning for both the datasets. This graph establishes that the time per round of active learning is limited to within 3 seconds. The address dataset takes longer since it has more unlabeled instances. The datasets used in this experiment are small and do not use any of the performance enhancements discussed in Section 2.1. This is a topic of our ongoing research.

4.2 Evaluating active learning on different base classifiers

In Figure 8 we plot the performance of active learning under three different classification methods. These graphs show that decision trees provide the best F accuracy overall. In the legend of Figure 8 we show the precision and recall values at the last round of active learning. D-trees dominate SVMs which in turn dominate NB in both the precision and recall values. However, D-trees show a much larger fluctuation in accuracy in the initial stages. This is to be



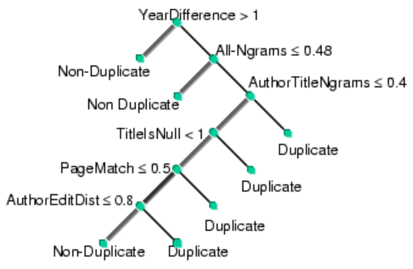
(a) Bibliography data



(b) Address data

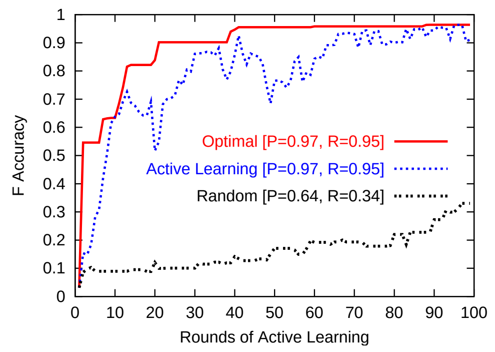
Figure 8: Comparing performance of different classification methods with active learning.

expected because decision trees are known to be *unstable* classifiers. For the address dataset, SVMs are better in the initial stages of active learning when the training data is small but they loose out later. This does not imply that for a fixed training set SVMs would be worse than D-trees. In these graphs we are evaluating a classifier both on its capability to return meaningful uncertainty values and its overall accuracy. SVMs are known to excel on accuracy but the uncertainty value measured as distance from SVM separator is perhaps not too meaningful. D-trees turn out to be better in the combined metric. This is good news because D-trees also offer other advantages of interpretability and indexability as discussed in Section 2.2. For example, we show below the final C4.5 d-tree output for the bibliography data. Such an output is easy to interpret and tune.

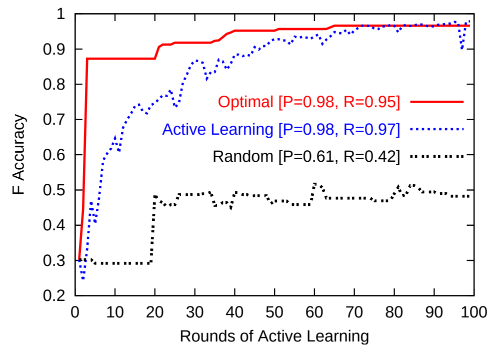


4.3 Comparing active learning with random selection

We evaluate the overall performance of active learning by comparing its speed of convergence to the peak accuracy with that of a random selection of the same number of instances in Figure 9. The graphs show three lines, one each for active learning, random, and optimal selection. We will discuss the comparison of active learning with random first. For both datasets, active learning shows clear superiority over random selection. Within just 100 of the more than 30,000 instances available, active learning is able to achieve a peak accuracy of 97% for the bibliography and 98% for the address dataset. The accuracy does not improve beyond these first 100 instances. The same number of instances selected randomly, achieve accuracy of just 30% and 50% respectively for the bibliography and address datasets. In fact, to achieve even 90% of the peak accuracy random selection needs 5600 instances for the address data and 2700 instances for the bibliography data.



(a) Bibliography data



(b) Address data

Figure 9: Speed of convergence with active learning, random selection and optimal selection

Another interesting observation from these experiments is that in the first 100 selected instances duplicates form 44% of the total for both data sets — a jump from the less than 0.5% fraction duplicates in the original unlabeled pool. Does this mean that the primary gain of active learning is due to correcting the extreme skewness in the original data? Or, are the particular set of instances important? We performed a second experiment by randomly selecting 100 instances but this time keeping the number of duplicates the same as after active learning. This yielded an average accuracy of only 40% on the bibliography data and 31% on the address dataset.

These numbers are important. They confirm our original intuition that manually collecting large *number of duplicates* will not achieve high accuracy unless proper care is taken in selecting a *confusing* enough set of non-duplicates to go with it. This is hard not only because the number of non-duplicates is large but also because it is not easy to know what non-duplicate would be misclassified as a duplicate with an existing training set.

4.4 Comparison with optimal selection

Another important question is how close our active selection method is to some absolute best method. We designed an *optimal* method that *knows the labels of all instances in our unlabeled set D_p through an oracle.* At each round of active learning, it then picks one instance ($n = 1$) as follows:

1. For each instance x in D_p
   (a) Add x with its correct label to the current training data T and train a classifier C_x.
   (b) Compute accuracy a_x of C_x in predicting class-labels of instances in $D_p - x$
2. Pick the instance x for which accuracy a_x was the highest.

This is the best algorithm one can design in the one-instance-at-a-time category of algorithms. This does not guarantee to give us the best subset of k instances for a fixed training size k, it just ensures optimality at each step where we pick one instance at a time. In Figure 9 we plot the accuracy of this optimal approach. For both datasets we notice that our chosen criteria of instance selection is indeed very close to the accuracy provided by the optimal approach that unrealistically assumes that all labels are known. One major difference is that optimal is smooth and monotonic whereas with active learning the accuracy fluctuates. In the legend part of Figure 9 we show the precision and recall values after the last round. Along both these metrics separately active learning is close to the optimal approach.

5 Related work

Recently, there has been renewed interest in the database community on the data cleaning problem [26, 10, 25] comprising several aspects, including, data segmentation, deduplication, outlier detection, standardization and schema mapping. For the specific problem of deduplication, most recent work [12, 22] has concentrated on the performance aspects assuming that the deduplication function is input by the user.

The problem of deduplication has long been relevant for library cataloging — see [29] for a survey. [16, 13] concentrate on hand-coding deduplication functions for the bibliography domain. The deduplication problem is also of interest to the statistics community in organizations like Census Bureau [31, 32, 6]. Much effort has been spent in designing domain-specific similarity functions for Census datasets. The learning approach is restricted to one-shot conventional classification using logistics regression and naive Bayes. Some of our similarity functions have been inspired by this literature. However, none of these systems address the difficulty of collecting a good covering set of training instances to start with.

Our approach of learning the deduplication function interactively bears resemblance to interactive relevance feedback used to refine queries over text and multimedia content [3]. In relevance feedback the goal is to learn a relevance function which in most cases boils down to learning appropriate weights of a weighted distance function. The key difference between relevance feedback and active learning is the type of examples shown to the user for collecting feedback. In most relevance feedback systems the user is shown the top few most *relevant answers* in each round whereas in active learning fast convergence rests on showing the user *the most uncertain* answers.

In Section 3 we have already discussed the various ways of doing active learning. Active learning has been applied in several domains in the past, including, text classification [20, 30, 17], and information extraction [1]. We believe ours is one of the first attempts at using active learning to solving a large-scale, practically motivated problem.

6 Conclusion

Deduplication, a key operation in integrating data from multiple sources, is a time-consuming, labor-intensive and domain-specific operation. ALIAS is a novel approach to easing this task by limiting the manual effort to inputing simple, domain-specific attribute similarity functions and interactively labeling a small number of record pairs. We presented a careful evaluation of a number of non-obvious design tradeoffs to ensure that the active learning process is practical, effective and can provide interactive response to the user. The final deduplication function is designed to be easy-to-interpret and efficient to apply on large datasets.

We find that active learning requires one to two orders of magnitude fewer pairs to be labeled than random selection. Our experiments show that starting from a highly skewed unlabeled pool with less than 0.5% duplicates, we are surprisingly able to selectively sample 100-fold more duplicates than non-duplicates, making the skew 50%. Also, the specific set of non-duplicates that we pick are important. If the same number of non-duplicates are picked without active selection our accuracy drops to half. Finally, we find that our chosen approach is close to an optimal approach.

Future work include more extensive running time evaluation, design of better similarity indices, and aiding users in designing good attribute similarity functions.

Acknowledgments

This project was funded by the Ministry of Information Technology, India under the project "Mobile agents for collaborative distributed applications", 2001-2002. We would like to acknowledge the help of Mrs. Pratibha Joag of the National Informatics Corporation for providing us the address dataset and Dr. Nageswara Rao for his support and encouragement during the project.

References

[1] S. Argamon-Engelson and I. Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[2] V. R. Borkar, K. Deshmukh, and S. Sarawagi. Automatic text segmentation for extracting structured records. In *Proc. ACM SIGMOD International Conf. on Management of Data*, Santa Barabara,USA, 2001.

[3] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proc. of SIGIR*, pages 292–300, 1994.

[4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] S. Chaudhuri, V. Narasayya, and S. Sarawagi. Efficient evaluation of queries with mining predicates. In *Proc. of the 18th Int'l Conference on Data Engineering (ICDE)*, San Jose, USA, April 2002.

[6] W. Cohen and J. Richman. Learning to match and cluster entity names. In *ACM SIGIR' 01 Workshop on Mathematical/Formal Methods in Information Retrieval*, 2001.

[7] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[8] R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001. Software available from "http://www.idiap.ch/learning/SVMTorch.html".

[9] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[10] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C. Saita. Declarative data cleaning: Language, model and algorithms. In *Proc. of the 27th Int'l Conference on Very Large Databases (VLDB)*, pages 307–316, Rome,Italy, 2001.

[11] L. Gravano, Panagiotis, and H. V. Jagadish. Approximate string joins in a database (almost) for free. In *Proc. of the 27th Int'l Conference on Very Large Databases (VLDB)*, Rome,Italy, 2001.

[12] M. A. Hernandez and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[13] J. Hylton. Identifying and merging related bibliographic records. Master's thesis, MIT, 1996.

[14] V. S. Iyengar, C. Apte, and T. Zhang. Active learning using adaptive resampling. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedinmgs of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pages 91–98, N. Y., Aug. 20–23 2000. ACM Press.

[15] R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence*, pages 234–245. IEEE Computer Society Press, available from `http://www.sgi.com/tech/mlc/`, 1996.

[16] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[17] R. Liere and P. Tadepalli. Active learning with committees for text categorization. In *Proceedings of AAAI-97, 14th Conference of the American Association for Artificial Intelligence*, pages 591–596, Providence, US, 1997. AAAI Press, Menlo Park, US.

[18] A. McCallum, K. Nigam, J. Reed, J. Rennie, and K. Seymore. Cora: Computer science research paper search engine. http://cora.whizbang.com/, 2000.

[19] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[20] A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[21] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[22] A. E. Monge and C. P. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

[23] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[24] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, 1993. software available from `http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz`.

[25] V. Raman and J. M. Hellerstein. Potters wheel: An interactive data cleaning system. In *Proc. of the 27th Int'l Conference on Very Large Databases (VLDB)*, pages 307–316, Rome,Italy, 2001.

[26] S. Sarawagi, editor. *IEEE Data Engineering special issue on Data Cleaning.* `http://www.research.microsoft.com/research/db/debull/A00dec/issue.htm`, December 2000.

[27] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

[28] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Computational Learing Theory*, pages 287–294, 1992.

[29] S. Toney. Cleanup and deduplication of an international bibliographic database. *Information Technology and libraries*, 11(1):19 – 28, 1992.

[30] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, Nov. 2001.

[31] W. E. Winkler. Matching and record linkage. In B. G. C. et al, editor, *Business Survey Methods*, pages 355–384. New York: J. Wiley, 1995. available from http://www.census.gov/.

[32] W. E. Winkler. The state of record linkage and current research problems. RR99/04, http://www.census.gov/srd/papers/pdf/rr99-04.pdf, 1999.

[33] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD)*, 2001.

[34] T. Zhang and F. J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. 17th International Conf. on Machine Learning*, pages 1191–1198. Morgan Kaufmann, San Francisco, CA, 2000.