ALIAS*: An Active Learning led Interactive Deduplication System

Sunita Sarawagi
sunita@it.iitb.ac.in

Anuradha Bhamidipaty
anu@it.iitb.ac.in

Alok Kirpal
alok@it.iitb.ac.in

Chandra Mouli
mouli@it.iitb.ac.in

Indian Institute of Technology Bombay

Abstract

Deduplication, a key operation in integrating data from multiple sources, is a time-consuming, labor-intensive and domain-specific operation. We present our design of ALIAS that uses a novel approach to ease 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 describe how active learning is useful in selecting informative examples of duplicates and non-duplicates that can be used to train a deduplication function. ALIAS provides mechanism for efficiently applying the function on large lists of records using a novel cluster-based execution model.

1 Introduction

Deduplication is a key operation in integrating data from multiple sources. The goal of the ALIAS deduplication system is to automate the manual, time-consuming process of removing duplicates in large semi-structured lists.

The main challenge in this task is defining a robust deduplication function that can capture when two records refer to the same entity in spite of the various inconsistencies and errors in data. ALIAS automates this task by learning the function from examples of duplicates and non-duplicates.

Early experience on real-life datasets showed that the quality of the learnt deduplication function critically hinges on being able to provide a large *covering and challenging* set of examples that bring out the subtlety of the deduplication function. Finding such examples is not easy because real-life data often has unexpected kinds of inconsistencies between duplicates hidden apart in large record lists, and spotting them involves tedious quadratic searches.

ALIAS provides an innovative method of interactively discovering such challenging training pairs in large lists through the use of **active learning** [1]. In contrast to an ordinary learner that trains a classifier using a fixed set of training instances, an active learner can actively select from a large pool of *unlabeled* instances, a subset that when labeled by the user will provide the highest information gain to the learner. The challenge in active learning is to pick the smallest number of examples that the user needs to label to teach the learner how to separate the duplicates from the non-duplicates.

Our experience with real-life datasets showed that ALIAS reduced the number of labeled training pairs required to reach peak accuracy by two orders of magnitude. After labeling less than 100 pairs selected interactively, the learnt deduplication function achieved the peak accuracy which a randomly chosen set of pairs could not achieve even with 7000 pairs. ALIAS is based on a number of careful design decisions to ensure that the active learning process is practical and can provide interactive response to the user. The final deduplication function output by ALIAS is designed to be easy-to-interpret and efficient to apply on large datasets.

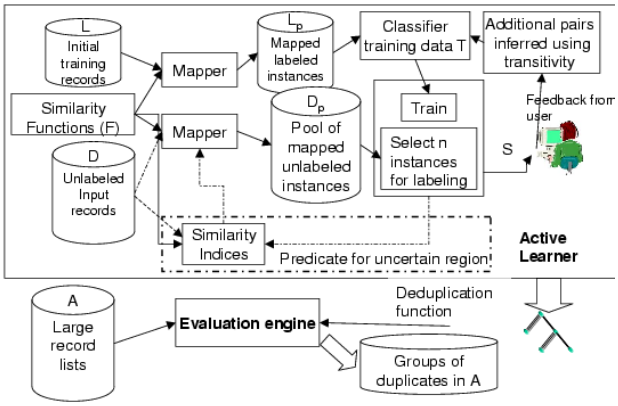
ALIAS provides a mechanism for optimizing a deduplication function so that it can be applied efficiently on very large lists without explicitly materializing all possible pairs of records. ALIAS uses a novel execution model consisting of operators that group records into clusters and process the clusters in various ways. This significantly limits the number of pairs that need to be materialized for exact evaluation.

2 Description of the system

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

*Active Learning led Interactive Alias Suppression

Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002



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.
   - Augment S with pairs inferred using transitivity of the "duplicates" relation.
   - Add augmented 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.

1. Database of records (D) The original set D of records in which duplicates need to be detected. The data has d attributes $a_1, \ldots a_d$, 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 five) 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 with n_f real-valued attributes and a class-label of either "0" or "1". These L_p instances are used to initialize a committee of N base classifiers to be used later for active learning.

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. 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 guaranteed to match all duplicates. For example, for citation data the year of publication could be one such grouping function or the first letter of the last name for address lists. Pairs are formed only within records of a group.
- Sampling: When we are trying to learn a deduplication function, we may not need to work on the entire set of records. 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. We support an alternative grouped sampling approach that hinges on being able to find a grouping function such as above.
- 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 $g(r) \geq C$

without loosing out on any qualifying record. This predicate can be evaluated via the index and later filtered for exact match.

However, we cannot exploit such similarity functions unless we modify our active learning mechanism that 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. ALIAS has an elaborate evaluation engine (described later) to efficiently process such predicates.

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) instances that it would most benefit from labeling. This is the responsibility of the active learner. The key insight here is to simultaneously build a few redundant classifiers using the available training dataset. The instances on which there is maximum disagreement amongst the predictions of the classifiers are the ones to be selected for labeling next. A sketch of the algorithm is given in Figure 2. More details can be found in [2].

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. This yields an additional set of n labeled instances. Since deduplication is a transitive relationship, these newly labeled pairs can be used to infer duplicate and not-duplicate relationship between other pairs of records. The augment set is 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.

Although, ALIAS is general enough to train any kind of classifier (Naive Bayes, Support Vector Machines, Decision trees and so on), we recommend using decision trees for their ease of interpretation and high accuracy. The output of a decision tree classifier is a set of if-then-else predicates on the similarity functions. ALIAS supports an elaborate evaluation engine for efficiently evaluating such functions on large lists of records. The evaluation engine uses several optimizations including,

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. P = predicate capturing the region where the N classifiers will likely disagree
4. I_p = subset of D_p that satisfies predicate P.
5. For each unlabeled instance x in I_p,
   (a) Find prediction $y_1 \ldots y_N$ from the N members.
   (b) Compute uncertainty or disagreement $U(x)$ as the fraction of the N predictions different from the majority prediction.
6. Return n instances by (weighted) sampling on the instances with the weight as $U(x)$.

Figure 2: Algorithm used by the active learning for selecting n instances for labeling

- reordering the similarity functions such that the less expensive ones are evaluated earlier than the expensive ones,
- delaying the explicit materialization of the cartesian product of large lists by operating on clusters of records
- using easy to evaluate filters before expensive similarity functions

More details of the execution model will appear in a paper in preparation.

3 Example sessions with ALIAS

We demonstrate the mechanism of interactively learning the deduplication function on data from examples of citation entries. We show how starting from a committee of single node decision trees learnt from a small number of initial training pairs, we progressively refine the tree as more pairs get labeled by the user to form the final deduplication function.

We start with two training pairs, one duplicate and another non-duplicate. In Figure 3 we show the first committee of three decision trees formed by the active learner using these two examples. The set of 14 input similarity functions are shown on the top left corner of the figure. The data has four fields: Author, Title, Year, Page number and All (the entire bib entry). The similarity functions are application of functions like ngrams match, word match and edit distance on

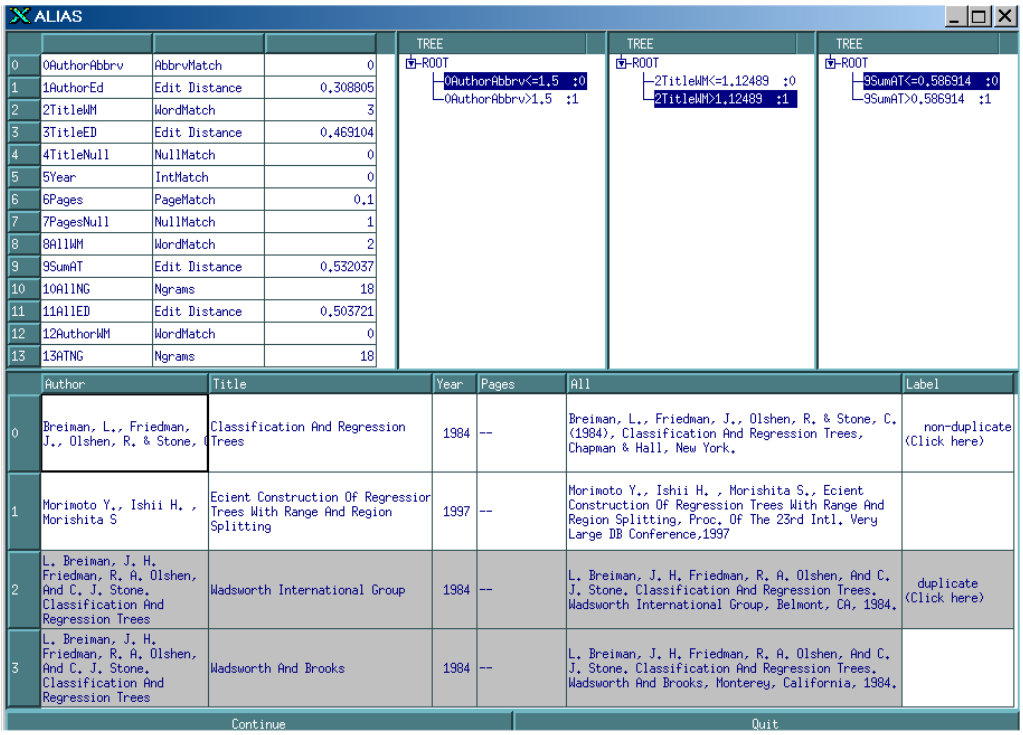


Figure 3: Initial stage of active learning after training on just two instances

Figure 4: The tree after training on 100 instances

a single text field or a concatenation of two or more of the text fields and, integer equality on the numeric fields like page number and year.

The three different trees use thresholds on three of these similarity functions to define the deduplication function. This shows the redundancy captured by the decision trees in the committee. The unlabeled instances that get conflicting predictions from these trees are selected by the active learner for labeling.

The bottom half of the figure shows two pairs of duplicates selected based on disagreement amongst the three trees. For the first pair, the figure shows the values of the similarity functions and highlights its position in the three trees. This pair is predicted as a non-duplicate by the first and third tree and duplicate by the second. The last column shows the predicted label. The user can correct these labels if they are wrong and submit them for inclusion in the training set.

As more and more examples get added to the training set, the trees gets refined. In Figure 4 we show an example refined tree obtained after about 100 tuples have been fedback through active learning. The user can edit the tree if he so chooses and save it for later use. Experimental results on the number of pairs to be labeled to converge to a peak accuracy appear in [2].

Acknowledgments

This project was funded by the Ministry of Information Technology, India under the project "Mobile agents for collaborative distributed applications", 2001-2002.

References

[1] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[2] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2002)*, 2002.