A Practical and Effective Sampling Selection Strategy for Large Scale Deduplication

Guilherme Dal Bianco, Renata Galante, Marcos André Gonçalves, Sergio Canuto, and Carlos A. Heuser

Abstract—The data deduplication task has attracted an considerable amount of attention from the research community in order to provide effective and efficient methods. The information provided by the user to tune the deduplication process is usually represented by a set of manually labeled pairs. In very large datasets, producing this kind of labeled set is a daunting task since it requires an expert to select and label a large number of informative pairs. In this article, we propose a two-stage sampling selection strategy (T3S) that selects a reduced set of pairs to tune the deduplication process in large datasets. T3S selects the most representative pairs by following two stages. In the first stage, we propose a strategy to produce balanced subsets of candidate pairs for labeling. In the second stage, an active selection is incrementally invoked to remove the redundant pairs in the subsets created in the first stage in order to produce an even smaller and more informative training set. This training set is effectively used both to identify where the most ambiguous pairs lie and to configure the classification approaches. Our evaluation shows that T3S is able to reduce the labeling effort substantially while achieving a competitive or superior matching quality when compared with state-of-the-art deduplication methods in large datasets.

Index Terms—Deduplication, signature-based deduplication

1 INTRODUCTION

We have witnessed a dramatic growth in the generation of information from a wide range of sources such as mobile devices, streaming media, and social networks. This has opened opportunities for the emergence of several new applications such as comparison shopping websites (e.g., Bizrate: www.bizrate.com/), digital libraries (e.g., CiteSeer: citeeex.ist.psu.edu/) and media streaming (NetFlix: www.netflix.com). These applications presuppose high quality data to provide reliable services.

However, data quality can be degraded mostly due to the presence of duplicate pairs with misspellings, abbreviations, conflicting data, and redundant entities, among other problems. For instance, a system designed to collect scientific publications on the Web to create a central repository (e.g., CiteSeer [17]) may suffer a lot in the quality of its provided services, e.g., search or recommendation services may not produce results as expected by the end user due to the large number of replicated or near-replicated publications dispersed on the Web (e.g., a query response composed mostly by duplicates may be considered as having low informative value). The ability to check whether a new collected object already exists in the data repository (or a close version of it) is an essential task to improve data quality.

Considerable improvements in data quality can be obtained by detecting and removing duplicates. Record deduplication aims at identifying which objects are potentially the same in a data repository [9], [14]. Although an old problem, it still continues to receive a significant amount of attention from the database community due to the inherent difficulty in producing a “replica-free” repository, especially in the context of large datasets.

A typical deduplication method is divided into three main phases: Blocking, Comparison, and Classification. The Blocking phase (aka the indexing phase) aims at reducing the number of comparisons by grouping together pairs that share common features [9]. A simplistic blocking approach, for example, puts together all the records with the same first letter of the name and surname attributes in the same block, thus avoiding a quadratic generation of pairs (i.e., a situation where the records are matched all-against-all). The Comparison phase quantifies the degree of similarity between pairs belonging to the same block, by applying some type of similarity function (e.g., Jaccard, Levenshtein, Jaro [14]). Finally, the Classification phase identifies which pairs are matching or non-matching. This phase can be carried out by selecting the most similar pairs by means of global thresholds, usually manually defined [3], [7], [24], [26] or learnt by using a classification model based on a training set.

In the case of large scale deduplication, the blocking and classification phases typically rely on the user to configure or tune the process. For instance, the classification phase usually requires a manually labeled training set. However, selecting and labeling a representative training set is a very costly task which is often restricted to expert users. Active learning approaches have been proposed to alleviate this problem by helping to select the most informative pairs [1], [4], [20]. Active learning can reduce considerably the number of pairs to be manually labeled when compared to random selection in order to produce competitive effectiveness.
Classifiers committees have been used in active learning to allow deduplication approaches to identify informative pairs based on divergences between the committee members [4], [20]. However, in the initial stages, these approaches still require a minimum training set (which is usually not small) and the definition of some thresholds to allow the classifiers to be learnt, thus still relying in considerable efforts from the experts.

In this context, we have successfully proposed the FS-Dedup framework [12], designed to select the “close-to-optimunm configuration” for large scale deduplication tasks with reduced user effort. A heuristic was proposed to select a balanced and informative set of candidate pairs to be labeled by the user to accurately identify the boundaries of the fuzzy region. The labeling effort is basically concentrated in a random selection of pairs inside the fuzzy region (the “challenging” pairs for the classification process). The proposed sampling within fixed similarity levels creates balanced subsamples, thus avoiding a sample selection bias while reducing the potential size of the training set. This allows more useful information to be obtained for the classification process in a faster pace. FS-Dedup was demonstrated to be more effective than manually tuned methods, while still reducing labeling efforts. However, the resulting subsamples may still be composed of redundant pairs, with negative impacts in the labeling effort.

In this article, we introduce a new step to our previous method aimed at reducing the redundancy in the subsamples, resulting in a new two-stage sampling selection (T3S) for deduplication, called T3S. Our proposed method is able to select a very small, non-redundant and informative set of examples with high effectiveness for large scale datasets. In more details, in the second stage a rule-based active sampling strategy, which requires no initial training set (as required in classifier committees), is incrementally applied to the selected subsamples to reduce redundancy. Further, we demonstrate that the two steps of our method are complementary, with mutual benefits for each other. While the second stage helps to remove redundancy, the first stage allows the second one to concentrate on the “most promising” portions of the search space for the most informative pairs to be labeled. As we shall see, just applying the rule-based active sampling on the entire dataset is computationally infeasible for large datasets. The final reduced training set produced by T3S is then integrated with our previous framework to efficiently identify the position of the fuzzy region and configure suitable strategies to classify the most ambiguous pairs.

We compare T3S with FS-Dedup as well as with two state-of-the-art active learning methods for deduplication [4], [20] and one unsupervised approach [8]. Experimental results obtained from synthetic and real datasets (one with about three million records) show that T3S reduces the training set size in about 16 times when compared to FS-Dedup and converges much faster to a high matching quality when compared to the baselines.

This paper is structured as follows. Section 2 reviews related work. Section briefly introduces some background concepts. Our proposed T3S approach is described in Section 4. Section 5 discusses the experimental results while Section 6 concludes the paper.

2 RELATED WORK

Record deduplication studies have offered a wide range of solutions exploiting supervised, semi-supervised, and unsupervised strategies. Supervised and unsupervised strategies rely on expert users to configure the deduplication process. The former assumes the presence of a large training set consisting of the most important patterns present in the dataset (e.g., [13], [25]). The latter relies on threshold values that are manually tuned to configure the deduplication process (e.g., [3], [7], [23], [26]). On the other hand, semi-supervised or active learning approaches, which are more closely related to T3S, have been used to reduce the user effort to configure the classification process. The goal of the active learning approaches is to select pairs from an unlabeled dataset which, when labeled, will bring more information gain to learn the classification model [14].

Traditional studies on active learning for binary classification are concerned with improving accuracy; in other words, they compute the classification quality on the basis of the number of pairs that are correctly classified [5], [11]. Such works cannot be straightforward applied to the deduplication task, since it is characterized by a high degree of imbalance (i.e., the non-matching pairs far exceed the number of matching pairs) and the metrics must be able to measure the fraction of true matching pairs that are recovered (i.e., precision and recall) [1], [4], [20]. For instance, [11] proposed the first general active learning approach for selecting pairs to be labeled where the classifier is least confident about the predictions. The authors of [16] exploit uncertainty among a committee of classifiers to define the pairs that will be labeled. Beygelzimer et al. [5] proposed an active learning approach, called IWAL, where the pairs are labeled on the basis of the divergence between the current hypothesis (i.e., the hypothesis that predicts the pair as matching) and an alternative hypothesis (i.e., the hypothesis that predicts the pair as a non-matching). The hypotheses are incrementally learnt by using the previously labeled pairs.

Committee-based strategies for deduplication, called ALIAS and Active Atlas respectively, are outlined in [20] and [22]. The committee identifies the most informative pairs to be labeled by the user as the unlabeled pairs that most classifiers disagree regarding their prediction. Active Atlas employs a committee composed by decision trees, while ALIAS uses randomized decision trees, a Naive Bayes and/or an SVM classifier. We included ALIAS as one of our baselines.

An alternative active learning method for deduplication was proposed in [1], where the objective is to maximize the recall under a precision constraint. The approach creates an N-dimensional feature space composed of a set of similarity functions, that are manually defined, and actively selects the pairs by carrying out a binary search over the space. However, the N-dimensional binary search may lead to a large number of pairs been queried, increasing the manual effort [4].

In [4], a strategy, referred as ALD, is proposed to map any active learning approach based on accuracy to an appropriate deduplication metric under precision constraints. This kind of approach projects a quality estimation of each classifier by means of points in a two-dimensional space. ALD conducts a binary search in this space to select the optimal
classifier that respects the precision constraint. The space dimensions correspond to the classifiers’ effectiveness, estimated by means of an “oracle”. The pairs used for training are selected by the IWAL active learning method [5]. Although the comparison is not completely fair since this active learning method uses a manually tuned blocking threshold, we included ALD as one of our baselines.

Corleone [18] aims at removing the expert intervention in crowdsourcing scenarios. It uses a committee of random forests to actively select informative pairs to be labeled. Divergence among trees is created using a randomization of parameters (similarly to ALIAS [20]). Differently from us, Corleone focuses on strategies to select a training set in the presence of noisy labeled pairs, a specific problem that happens when different users are responsible for labeling the pairs in a crowd. To address this, three heuristics to identify the stopping point for the learning process are proposed.

FS-Dedup [12] proposes a different approach to configure the deduplication process with a reduced user intervention. It sorts the pairs by means of the similarity value. The sorted set is divided up by using fixed levels (ranging across the similarity value). Then, a strategy is proposed to label only the random subsamples that are composed of the most ambiguous pairs inside each level. Additionally, strategies to configure the blocking phase and identify the classification threshold are proposed. However, FS-Dedup depends on subsamples that may include redundant information, representing a waste of manual effort. In contrast, our proposed T3S method applies the SSAR active learning (detailed in Section 3.3) incrementally in all the subsamples to produce the training set. T3S selects a small set composed of the most informative pairs and reduces the final training set size to a much greater extent than FS-Dedup, as we shall see.

Focusing on automatic training set construction, [8] proposed a two-step approach. In the first step, an initial training set is automatically produced by selecting examples among the most and least similar pairs. In the second, the initial training set is used to feed supervised classifiers and unlabeled pairs are iteratively classified, labeled and added into the training. We also included this approach as one of our baselines. Similarly, [6] proposed an approach to select a balanced training set by identifying the most and least similar pairs to be manually labeled. Differently, from above, the training set is complemented by using random selected pairs. In the same line, [19] proposed a generic deduplication platform along with strategies similar to those proposed in [6] to select the training set based on thresholds values. More specifically, a defined number of pairs with similarity higher and lower than a specified threshold is selected to be labeled by the user. Since these thresholds depend largely on the dataset dirtiness, it is very difficult, even for expert users, to define precise values.

3 Background Concepts

In this section, we present some background necessary to properly understand our proposed approach, as some of the used concepts are recent and non-trivial. First, we specify the main concepts behind Sig-Dedup algorithms adopted as deduplication core by our approach in the blocking and classification steps. Then, we explain the notion of fuzzy region which represents a subset composed of ambiguous pairs.

3.1 Signature-Based Deduplication (Sig-Dedup)

Sig-Dedup has been proposed to efficiently handle large deduplication tasks. It maps the dataset strings into a set of signatures to ensure that similar substrings result in similar signatures. The signatures are computed by means of the well-known inverted index method.

However, there are two drawbacks to produce indexes for deduplication tasks. The first is that all the tokens or sub-strings of each record must be evaluated. This evaluation produces candidate pairs with quadratic growth [26]. The second drawback is that the entire record must be analyzed before it can be incorporated into the index. This procedure can be expensive when the record is large.

Several studies have addressed the question of reducing the quadratic candidate generation [3, 7, 24, 26]. Chaudhuri et al. [7], for instance, proposed the prefix filtering, aimed at pruning out large number of unrelated pairs, indexing only the less frequent tokens of each record. More specifically, the preprocessing phase tokenizes each record using words or N-Grams to create token sets. In contrast with the performance of word tokenization, N-Gram tokenization can improve the effectiveness of datasets with short character mistakes. However, it produces inverted index structures with a vast number of tokens and large blocks, which increases the computational demands. In the indexing phase, each token set is sorted by following the global frequencies (global ordering $\theta$). The purpose of the sorted set, called signature vector, is to establish which tokens are rarer in the dataset and which are more frequent (e.g., stop words). In the indexing phase the less frequent tokens are indexed, with only the prefix of each signature vector being selected. This produces blocks with fewer records, reducing the number of candidate pairs. The prefix filtering is formally defined below:

Definition 1. Assume that all the tokens in each record are ordered by a global ordering $\theta$. Let $p$-prefix of a record be the first $p$ tokens of the record. If $\text{Jaccard}(x,y) \geq t$ then the $(p)$-prefix of $x$ and $(p)$-prefix of $y$ must share at least one token. [7]

In addition to prefix filtering, length filtering can also be applied to remove records whose length variation is higher than a specified threshold [3]. Xiao et al. [26] observed that after the insertion of the prefix filtering, the number of candidate pairs continues to increase with quadratic growth. Thus, two additional filters are proposed to reduce the number of candidate pairs. The first, called positional filtering, checks the variations of the token position in the prefix of the signature vector. The second, called suffix filtering, checks if the candidate pairs have variations in position, by using a recursive binary search in the suffix of the signature vector. On the other hand, [23] observe the potential of the Sig-Dedup algorithms and provide a MapReduce extension to handle large datasets with reduced time.

Overall, Sig-Dedup approaches represent the state-of-the-art of large scale deduplication. Bayardo et al., for instance, efficiently deduplicated datasets with millions of records [26] using Sig-Dedup. Nevertheless, studies that focus on Sig-Dedup approaches overlook the problem of selecting the best
configuration since, even for an expert user, the number of possibilities to define the threshold values is usually very large. Our two-stage sampling selection approach focuses on reducing such user effort, allowing a small and representative candidate set to be produced, which can provide the information required to identify these thresholds.

3.2 Fuzzy Region

In [15] the first statistical foundations of the deduplication problem were developed [10]. The method seeks to classify pairs in a product space (A × B) from two sets of records (A and B) into: M represents the set of matching pairs, U represents the set of non-matching pairs, and P represents the set where matching and non-matching pairs co-exist. The pairs in the M set typically share common characteristics. The pairs in the U set usually have isolated agreements of the characteristics. Set P contains pairs with a high degree of ambiguity and requires human intervention to classify them. Because of this, we called the set P the fuzzy region.

Some types of functions, called agreement patterns (γk), are applied to determine whether the characteristics of a pair of records (ai, bi) represent a matching pair or not. A simplistic agreement pattern establishes if the fields have exactly the same value. A probabilistic model assigns a weight W for each γk, i.e., \( w_k = \frac{\ln(m_k)}{\ln(u_k)} \) if \( \gamma_k = 1 \) and \( w_k = \frac{\ln(1-m_k)}{\ln(1-u_k)} \) if \( \gamma_k = 0 \), where \( m_k \) and \( u_k \) represent the conditional probabilities that observes whether the characteristic k is matching or non-matching. Weight \( w_k \) has a positive value for a matching pair and, otherwise, a negative value. A decision model is devised by computing the sum of the weights \( w_k, w_{k+1}, \ldots, w_n \), and comparing this value against two threshold values \( \alpha \) and \( \beta \). An optimal decision rule that is given by:

- if \( \sum w_{k,n} < \alpha \), the pair is inserted into set U;
- if \( \alpha < \sum w_{k,n} < \beta \), the pair belongs to set P;
- if \( \sum w_{k,n} > \beta \), the pair is inserted into set M;

The main challenge of this model is to determine the optimal values for \( \alpha, \beta, m_k \) and \( u_k \), which minimize the set P (reduce the clerical review). In large scale deduplication, the fuzzy region may contain a large number of pairs, which means it is not feasible to employ human judgment, as suggested by [15]. The manual definition of both boundaries \( \alpha \) and \( \beta \) may also be imprecise. Our Two-Stage Sample Selecting strategy is able to provide a more practicable identification of the matching pairs inside the fuzzy region, by using only a reduced set of pairs to be manually labeled.

4 TWO-STAGE SAMPLING SELECTION (T3S)

In this section, we outline our proposed two-stage sampling selection aimed at selecting a reduced and representative sample of pairs in large scale deduplication. We integrate T3S with our previous FS-Dedup framework to reduce the user effort in the main deduplication steps (e.g., blocking and classification). Fig. 1 briefly illustrates how the T3S steps work together in a collaborative way. First, a strategy is employed to identify the blocking threshold, and thus produce the candidate pairs (Fig. 1(1)), as introduced in [12]. The dotted box represents the main steps of T3S. In its first stage, T3S produces small balanced subsamples of candidate pairs (Fig. 1(2)). In the second stage, the redundant information that is selected in the subsamples is removed by means of a rule-based active sampling [21] (Fig. 1(3)) which requires no previously labeled training set. Following this, we describe how these two steps work together to detect the boundaries of the fuzzy region (Fig. 1(4)). Finally, we describe our two classification approaches, also introduced in [12], which are configured by using the pairs manually labeled in the two stages (Fig. 1(5)).

4.1 Identifying the Approximate Blocking Threshold

In this step, the approximate blocking threshold is determined by using the Sig-Dedup filters (e.g., regarding the number of tokens that must be used) that maximize recall, i.e., that minimize the chance of pruning out actual matching pairs. We call this blocking threshold the initial threshold. Ideally, the set of candidate pairs produced using the initial threshold contains all the matching pairs.

As this step is performed without user intervention, we rely on generalizations as a means of becoming closer (or making an approximation) to the ideal scenario. In fact, the number of true matches and non-matches is not known a priori, but the initial thresholds are defined in order to minimize the number of “lost” matching pairs that are outside the interval for analysis. The other steps of our method are used to prune out the non-matching candidate pairs. It should be stressed that, also to avoid user intervention, the initial threshold represents a single global threshold for all the blocks.

It is worth noting that the set of candidate pairs is produced using the Sig-Dedup filters (i.e., prefix, length, position, and suffix filtering) and that these are configured with the initial threshold. The main purpose of this threshold is to define how many tokens are indexed by the sorted record (i.e., the records are resorted using the global frequency of tokens). Notice that in this step, the similarity value of each pair is not used to prune out pairs since we do not know the exact threshold value that is able to discard non-matching pairs. Additionally, we assume that in large datasets (composed of several millions of records) the number of matching pairs represents a small subset of the dataset. For instance, two identical datasets (e.g., two millions of record from each dataset) that are matched must have less matching pairs than the total number of records in such datasets. However, this generalization does not hold when a record has several thousand of duplicates and produces true pairs almost quadratically. This means that the dataset is largely composed of redundant information, which is unexpected in real data repositories.

In large datasets, it may not be feasible to run the Sig-Dedup filters with different thresholds due to the high computational costs. In light of this, we propose a stopping
criterion to estimate the initial threshold. A random subset is selected from the dataset that is matched by using a variable threshold which varies in fixed ranges. The stopping criterion specifies that the number of pairs needed to satisfy the Sig-Dedup filters must be lower than the subset size. When compared with the entire dataset, the random subset naturally decreases the number of true matching pairs. Thus, if the Sig-Dedup filters create more candidate pairs than the input set, there is a clear sign that the frequent tokens have been indexed. When the threshold value is incrementally increased, fewer tokens in the sorted record are indexed, thus reducing the number of candidate pairs. On the other hand, a high threshold value selects few tokens in the sorted record and a lot of matching pairs can be pruned out. The stopping criterion produces a threshold that avoids both: a large generation of candidate pairs and recall degradation. This notion is formalized below.

**Definition 2.** Consider a subset \( S \), created from a randomly sampled dataset \( D \) and a range of thresholds with fixed step \( \theta_j = 0.2, 0.3, \ldots, \) and 0.9. \( ^1 \) The subset \( S \) is matched using each threshold value \( \theta_j \). The initial threshold will be the first \( \theta_j \) that results in a number of candidate pairs smaller than the number of records in \( S \).

After defining the global initial threshold value for the blocking process, the entire dataset is matched to create the set of candidate pairs. It is important to notice that the initial threshold does not represent an optimal one but it is a threshold able to avoid a quadratic generation of pairs while avoiding to miss true matches. This step focuses on recovering (almost) all pairs with high likelihood of representing a matching in large datasets. The false-positive pairs produced by the initial threshold value can be pruned out in the next steps (i.e., fuzzy region identification and classification).

At the end, these candidate pairs are sorted using their similarity values to produce a ranking. In the next step, using this ranking it is possible to identify the pairs with the highest (true matching pairs) and lowest similarities (non-matching pairs). This step represents a strategy to generate candidate pairs and categorize them. It makes it easier to select a specific pattern of pairs, i.e., highly positive or highly negative candidate pairs.

### 4.2 First Stage: Sample Selection Strategy

We outline our proposed sample selection strategy to produce balanced subsamples of candidate pairs. The main idea of the first stage is to discretize the ranking (produced in the previous step) so that small subsets of candidate pairs can be selected to reduce the computational demand of the T3S second stage. A simplistic approach to produce samples might be to select random pairs within the set of candidate pairs. However, as the set of pairs is basically formed of non-matching pairs (most pairs in the dataset do not correspond to replicas\(^2\)), this kind of approach will result in samples having low informativeness. In this way, the second stage of T3S will hardly be able to select representative samples, because of the lack of informativeness in the matching pairs.

The first stage of T3S adopts the concept of levels to allow each sample to have a similar diversity to that of the full set of pairs. The ranking, created by the blocking step, is fragmented into 10 levels \((0.0-0.1, 0.1-0.2, 0.2-0.3, \ldots, \text{and } 0.9-1.0)\), by using the similarity value of each candidate pair. Two main reasons explain our choice for ten intervals. First, as shown in a previous characterization of the datasets \([12]\), the effectiveness of the deduplication process declines smoothly after the optimal threshold value is found and 10 intervals is enough to capture such behavior and define the needed thresholds. Second, an increase in the number of intervals (using for instance, a 0.05 step, i.e., 0.80-0.85) could substantially increase the number of candidate pairs that would have to be analyzed by our method (T3S) to identify the thresholds, as the levels would become more ambiguous. This would force a more precise identification of the fuzzy region boundaries implying in more labeled pairs to minimize the error rate. As we shall see in our experiments, a small number of good choices (as low as 30 pairs) is enough for our goals and this increase in effort much probably would not be worth it.

Inside each level, we randomly select candidate pairs to create samples with of an empirically defined size. For instance, all pairs with similarity within the \([0.0-0.1]\) interval are inserted into the first level. This fragmentation produces levels composed of different matching patterns to prevent non-matching pairs dominating the sample. For instance, level \([0.0-0.1]\) is composed of a large number of non-matching pairs (i.e., highly dissimilar records) while level \([0.9-1.0]\) has matching pairs only. Thus, it is possible produce a balanced set of pairs to be used in T3S next stage to remove redundancy in the information randomly selected.

### 4.3 Second Stage: Redundancy Removal

The first stage produces samples by carrying out a random selection of pairs inside each level. As we observed in \([12]\), the subsamples are an effective means of detecting the fuzzy region boundaries, especially when the size of the level is quite large. However, several pairs selected inside each level are composed of redundant information which does not help to increase the training set diversity (i.e., they waste manual labeling effort).

The second stage of T3S aims at incrementally removing the non-informative or redundant pairs inside each sample level by using the SSAR (Selective Sampling using Association Rules) active learning method \([21]\). By redundant, we mean pairs carrying very similar information; the inclusion of a redundant pair in the training set for the classification step does not contribute with useful information for the learning process. Yet, these pairs can belong to different levels. For example, two pairs with similarity value of 0.29 and 0.31 belong to levels \([0.3-0.4]\) and \([0.4-0.5]\), respectively (as illustrated in Fig. 2). The purpose of SSAR is to select for labeling only the most informative pairs required to maximize the training size diversity while minimizing labeling effort.

When compared to the mentioned active learning approaches (e.g., \([1], [4]\)), SSAR has several advantages such as: (1) not requiring an initial labeled set, as needed by

---

1. We avoid using the 0.1 threshold since this would imply in an enormous amount of comparisons. Empirically, we have observed that the initial thresholds are always higher than 0.2.
2. Blocking only works to avoid unnecessary comparisons, but even within a block, most candidates still do not correspond to replicas.
approaches based on committees; (2) having a clear stopping criteria, a property that many approaches do not possess; and (3) the capability of selecting very few but very informative instances on an informativeness criteria grounded on lazy association rules. More specifically, SSAR selects an unlabeled pair \( u_i \) for labeling by using inferences about the number of association rules produced within a projected training set specific for \( u_i \). The projected training set is produced by removing from the current training set \( D \) instances and features that do not share features values with \( u_j \). When compared with the current training set, the unlabeled pair with less classification rules over the projected training set represents the most informative pair. A detailed example of this part of the SSAR algorithm is shown in Fig. 3.

Algorithm 1. SSAR: Rule-based Active Selective Sampling

```plaintext
Require: Unlabeled set \( T \) and \( \sigma_{\text{min}} \) (\( \approx 0 \))
Ensure: The training set \( D \)
1: while true do
2:   for all \( u_i \in T \) do
3:     \( D_{ui} \leftarrow D \) projected according to \( u_i \)
4:     \( R_{ui} \leftarrow \) extract useful rules from \( D_{ui} \)
5:   end for
6:   if \( D = \emptyset \) then
7:     \( \lambda_{ui} \leftarrow u_i \) such that \( u_i \) is the most representative item of \( T \).
8:   else
9:     \( \lambda_{ui} \leftarrow u_i \) such that \( \forall u_j : |R_{ui}| \leq |R_{uj}| \)
10:    end if
11:   if \( \lambda_{ui} \in D \) then
12:      break
13:   else
14:      LabelPair(\( \lambda_{ui} \))
15:      \( D \leftarrow D \cup \{ \lambda_{ui} \} \)
16:    end if
17: end while
```

Details of SSAR are shown in Algorithm 1. At each round, an unlabeled pair \( u_i \) is used as a filter to remove irrelevant features and examples from \( D \). In other words, the projected training data \( D_{ui} \) is obtained after removing all the feature-values that are not present in \( u_i \) (line 3). Next, a specific classification rule-set \( R_{ui} \) is extracted from \( D_{ui} \) (line 4). The number of rules created by each projected set represents its informativeness. The objective of this procedure is to select the most dissimilar unlabeled pair by making a comparison with the current training set. Unlabeled pairs composed of a considerable number of common features compared with the current training set produce a large number of rules, showing that they provide low information gain. On the other hand, an unlabeled pair composed of dissimilar features produces a projection which generates few rules (line 9). If the most dissimilar pair is not already present in the training set, it is labeled by the user and inserted into the training set \( D \) (lines 14 and 15). After this, a new round is performed and the training set must be re-projected for each remaining unlabeled pair to determine which one is most dissimilar when compared to the current training set. If the selected pair is already present in the training set (lines 11-13) the algorithm converges, assuming that the training set has the same degree of diversity as the unlabeled dataset. In the beginning of the algorithm, when the training set \( D \) is empty, SSAR selects the pair that shares most feature values with all other unlabeled pairs to initially compose the training (line 7). The idea is that this pair is the best "representative" of the information contained in the collection.

The main SSAR drawback is the high computational cost of re-projecting a large unlabeled dataset at each round. The computational complexity is \( O(S \cdot |U| \cdot 2^m) \), where \( \text{"S"} \) is the number of pairs selected to be labeled, \( |U| \) represents the total number of candidate pairs (reminding that rules have to be generated for all pairs in the unlabeled set) and \( "m" \) is the number of features (the cost of the rule generation is proportional to the number of features.). In other words, \( |U| \) pairs must be re-projected each time that a labeled pair is attached to the current training set, producing a computationally unfeasible time to process large datasets. For example, in a synthetic dataset with 105,000 records the SSAR takes about 13 hours to perform, while in a dataset with 150,000 records it takes about 31 hours.\(^3\) In other words, if we increase the dataset size by almost 50

\(^3\)This experiment was conducted by using synthetic datasets on a quad core 2 Duo 2.4 GHz machine with 4 GB of RAM.
percent, SSAR increases its runtime 2.3 times. Our sample selection strategy allows SSAR to perform in large datasets, reducing substantially the universe of \(\#[\text{U}]/\#\) pairs into levels of predefined size (i.e., 10...1,000 pairs).

The T3S first stage provides exactly the right conditions to allow SSAR to perform in large datasets by producing samples of a reduced size at each level. A naive method would be to produce the training set that is sent to a single level each time to be performed by SSAR. The final training set is created by joining together the subsamples of each level. Nevertheless, SSAR may select pairs with similar information at the border of the levels by producing a training set with redundant information. For instance, as Fig. 2 illustrates, the levels [0.1-0.2, 0.2-0.3,..., and 0.5-0.6] have pairs at the boundaries with similar and highly redundant information; thus if a single level is performed independently each time, redundant pairs will still be inserted in the training set.

Our sample selection strategy incrementally invokes SSAR by using each level and the current training set as input. More specifically, the first level [0.0-0.1] is actively selected by SSAR using an empty training set as input. T3S starts at the lower levels to select the matching pairs that carry more information (i.e., true matching pairs with low similarity). As “easy” and redundant non-matching pairs tend to be similar, more dissimilar true matching pairs may be privileged by the SSAR selection criteria. These are the most difficult and ambiguous cases that we really want to include in the training set as pairs with high similarity are easy to identify. After this, the levels [0.1-0.2,...,0.9-1.0] are inspected by SSAR by using the current training set in the previous levels as input. It is expected that SSAR will select more pairs at the first level than in the following levels as: (1) the training set is initially empty; and (2) the first level has several times more candidate pairs than the other levels (i.e., this subsample is composed of a greater feature diversity than the other levels). As Fig. 2 shows, by using the incremental active selection, the redundant pairs at the levels [0.1-0.2, 0.2-0.3, 0.3-0.4, 0.4-0.5, and 0.5-0.6] can be removed, reducing the labeling effort.

Finally, it should be stressed that the incremental selection of pairs creates samples that are composed of complementary information. In other words, the incremental selection of pairs depends on the information selected in the previous levels to produce the training set. When more pairs are labeled by the user, the training set becomes more informative and only the most dissimilar pairs are selected by SSAR.

### 4.4 Detecting the Fuzzy Region Boundaries

In this section, we detail how the training set created by the two stages of T3S is able to detect the fuzzy region boundaries. We describe in detail the proposed approach for detecting the fuzzy region:

**Definition 3.** Let Minimum True Pair-(MTP) represent the matching pair with the lowest similarity value among the set of candidate pairs.

**Definition 4.** Similarly, let Maximum False Pair-(MFP) represent the non-matching pair with the highest similarity value among the set of non-matching pairs.

The fuzzy region is detected by using manually labeled pairs. The user is requested to manually label pairs that are selected incrementally by the SSAR from each level. However, the pairs labeled by the user may result in MTP and MFP pairs which are far from the expected positions, as specified in Definitions 3 and 4. To minimize this problem, we assume that the levels to which the MTP or MFP pairs belong are defined within fuzzy region boundaries. For instance, if the MTP and MFP values are 0.35 and 0.75 respectively, all the pairs with a similarity value between 0.3 and 0.8 belong to the fuzzy region. We call the fuzzy region boundaries \(a\) and \(b\).

Algorithm 2 identifies the fuzzy region boundaries by using the T3S strategy. First, SSAR is invoked to identify the informative pairs incrementally inside each level to produce a reduced training set (lines 2-5). The pairs labeled within each level are used to identify the MFP and MTP pairs. The pair labeled as true that has the lowest similarity value defines the MTP (line 8), then, the following levels are analyzed to identify the non-matching pair with the highest similarity value (line 12).

**Algorithm 2.** Active Fuzzy Region selection

```
1: i = 0; MFP = Null; MTP = Null; trainingSet = NULL;
2: for i = 0 → 10 do
3:   trainingSet = SSAR(L, trainingSet)
4:   l = i + 1
5: end for
6: for i = 0 → 10 do
7:   if L does not contains only false and MTP = Null
8:   MTP = SelectLowestTruePair(LP);
9: end if
10: if L does not contains only true and MTP! = Null
11: MFP = SelectHighestFalsePair(LP);
12: end if
13: return MTP, MFP and LP;
```

In our previous work (FS-Dedup [12]), a strategy was employed to select the subsamples to be labeled looking for the fuzzy region boundaries. In an extreme case, where the fuzzy region depends on the label for most of the levels, a large number of redundant information will be manually labeled. T3S already assumes that all the subsamples may contain useful information and all of them are performed incrementally to select only dissimilar pairs (those that will bring more information gain to the process). It should be noted that the information that can be used at the lowest levels to identify the MTP represents the most dissimilar pairs. In this scenario, the large number of non-matching pairs that are present at this level are highly redundant and not informative to identify the fuzzy region boundaries. Thus, our strategy is mainly concerned with the selection of the dissimilar pairs, which are exactly the most informative means of identifying the \(a\) and \(b\) values.

The similarity value of the MTP and MFP pairs identifies \(a\) and \(b\) values. The fuzzy region is created by all the
candidate pairs with a similarity value between $\alpha$ and $\beta$. After the boundaries have been defined, the pairs belonging to the fuzzy region are sent to the Classification Step. This process considerably reduces the number of candidate pairs since pairs below the fuzzy region can be pruned out. Similarly, the set above $\beta$ is directly considered as matching pairs by the deduplication process, being not further analyzed. After the MTP and MFP have been identified, the fuzzy region is sent to the classification step along with the pairs that are manually labeled by the user.

4.5 Classification Step
The Classification step aims at categorizing the candidate pairs belonging to the fuzzy region as matching or non-matching. We use two classifiers in this step (as introduced in [12]): T3S-NGram and T3S-SVM. T3S-NGram maps each record to a global sorted token set and then applies both the Sig-Dedup filtering and a defined similarity function (such as Jaccard) to the sets. The token set does not consider the attribute positions, by allowing an exchange of attribute values. The drawback of T3S-NGram is that different attributes are given the same importance (i.e., they have the same weights). In other words, an unimportant attribute value with a large length may dominate the token set, and lead to distortions in the matching. On the other hand, T3S-SVM assigns different weights to different attributes of the feature vector, by using the SVM algorithm, based on their relative discriminative power. However, there is not an unique and globally suitable similarity function that can be adapted to different applications [25], and this makes it difficult to configure the method for different situations. Moreover, long text attributes can be mapped to non-appropriated feature values causing a loss of information in the classification process. As both methods have advantages and drawbacks, we make use of both of them in our T3S framework.

The set of pairs used to train the classifiers is selected using our proposed two-step approach and labeled by the expert. T3S produces a highly informative and more balanced set of positive and negative pairs that is used for both: to feed the classification algorithm and to identify the fuzzy region position.

T3S-NGram is designed to use the labeled sample to detect where the matching and non-matching are concentrated so that the threshold that removes the non-matching pairs can be selected. Thus, yet another specific threshold is required to identify the matching pairs inside the fuzzy region using the NGram tokenization: the NGram Threshold.

In the following section, we describe the strategy that can be employed to automatically identify the NGram Threshold.

First, the similarity of each labeled pair is recomputed by means of a similarity function along with the NGram tokenization. After this, the labeled pairs are sorted incrementally by the similarity value and a sliding window with fixed-size $N$ is applied to the sorted pairs. The sliding window is relocated in one position until it detects the last windows with only non-matching pairs. Finally, the similarity value of the first matching pair encountered after the last windows with only non-matching pairs, defines the NGram threshold value.

4. SVM exploits a quadratic optimization procedure whose output is the relative discriminative power of each attribute.

It should be pointed out that we can only perform this procedure because we have the labels of the pairs given by the user in the previous step. T3S is configured with the Ngram threshold value so that it can be applied to all the fuzzy region pairs. Finally, the candidate pairs that survive the filtering phase and meet the Ngram threshold value are considered as matching ones.

5 Experimental Evaluation
5.1 Datasets
We use both synthetic and real datasets to evaluate our framework. The real datasets are created by merging two different real datasets in the same domain to produce a deduplication scenario. As the real datasets do not have gold standards, we also use synthetic datasets to create controlled scenarios in order to better evaluate the methods. Since all solutions are compared under the same conditions, we believe our experimentation is fair.

We create synthetic datasets by using the Febrl Dsgen tool [10] as it is employed experimentally in several other works [8], [13]. This generator works by first creating the original records based on real-world vocabularies (e.g., names, surnames, street name, among others). After this, the original records are randomly changed (e.g. merging words, inserting character mistakes, and deleting attributes) to create the matching pairs. Errors are based on noise patterns found in real data. We simulate three scenarios: the clean dataset contains 5 percent of duplicates; the dirty-clean dataset contains 20 percent; and the dirty dataset contains 50 percent of duplicates. The records are synthetically built with 10 attributes: first name, surname, age, sex, address, state, street number, phone number, date of birth, and social security number.

Since large scale datasets are not easily available with gold standards, we manually create two real ones. The first one, called IMDBxNetflix, was created by accessing the public APIs of two movie databases: IMDb: www.imdb.com and NetFlix. The IMDB dataset has more than a million records, and the publicly available Netflix dataset contains about 160,000 records. Netflix focuses on specific market information, and hence cannot be considered as a “pure” subset of IMDB movies. Only a small fraction of both datasets represents matching pairs. We integrate the datasets using the common attributes: title, director, and year of release.

The second real dataset—DBLPxCiteeseer—is formed by merging DBLP: www.informatik.uni-trier.de/~ley/db/ and Citeeseer. Both are digital libraries storing information about scientific (Computer Science) publications. The Citeeseer dataset indexes new entries by automatically crawling web pages [17]. In this kind of scenario, the Citeeseer entries are affected by different levels of noise, for example, incomplete attributes, different name conventions, and other impurities. In contrast, DBLP entries are manually inserted, resulting in higher quality. We produced the DBLPxCiteeseer dataset by using the attributes title, author, and publication year. Table 1 summarizes the synthetic and real dataset configurations.

As the real datasets do not have gold standards, we asked five users (Computer Science graduate students) to label a training set composed of 100 pairs sampled from level 0.1-0.2 until level 0.9-1.0, resulting in around 900 pairs.
We did not include level 0.0-0.1 because matching pairs are extremely rare at this level. We created the test set manually for both datasets by evaluating all candidate pairs from the years 1988 to 1990 [2]. This resulted in 3,009 and 3,137 pairs in the DBLPxIMDB and IMDBxNetFlix datasets, respectively. The pairs with incomplete or ambiguous information were labeled by using additional information (e.g. the respective websites, metadata, among other types of information). The datasets are available for download at http://www.inf.ufrgs.br/~gbianco/datasets.

5.2 Evaluation Metrics
We used standard measures such as precision, recall and F1. For the synthetic datasets, we ran each experiment 10 times, reporting the average rates and standard deviation. All results were compared with statistical significance tests (paired t-test) with a 95 percent confidence interval.

5.3 Experimental Setup
The Sig-Dedup implementation PPJoin++ [26] was used as the deduplication core to run T3S and FS-Dedup. We configured PPJoin++ with Jaccard as the similarity function. We also used the active sampling implementation (SSAR) described in [21].

For FS-Dedup, we experimentally determined the sliding window size of two for synthetic datasets and five for the real datasets as the best ones. All true matching pairs are known in the synthetic datasets, resulting in a small sliding window. On the other hand, the gold standard created by the users for the real datasets may be susceptible to labeling mistakes. Thus, we set a sliding window value of five in these datasets to avoid distortions in the classifier configuration. As T3S has a reduced training set, it is natural that all the datasets use a small sliding window value of two.

As previously discussed, T3S uses the similarity degree of each pair to produce a ranking. However, as there is no universal similarity function good for all cases, different functions may be used with a same field to increase the number of features to be compared and, potentially, the quality of the training set selected by T3S. Thus, we evaluated T3S with two sets of pairs. The first set was produced by using only one similarity function (e.g. a dataset composed of three fields has three features), which we called T3S(1SF). The second set was produced by using two different similarity functions (e.g. the previous dataset would be composed of six features), which we called T3S(2SF). For the similarity functions we used standardized Levenshtein and Jaccard.

We used the libSVM package: www.csie.ntu.edu.tw/~cjlin/libsvm for T3S-SVM and Sig-Dedup-SVM as implementation of SVM. We used a RBF kernel and the best γ and cost parameters were obtained through cross-validation in the training set. The features were created by computing NGram-based Jaccard similarity which is known to be efficiently computable [14].

We consider four baselines: (1) our proposed previous framework, referred as FS-Dedup [12]; (2) ALD [4]; (3) ALIAS [20]; and (4) an automatic training set selection method proposed in Christen [8]. In case of ALD, an implementation is available by request. We defined the precision threshold value as 0.85 and the oracle was defined with 30 pairs, as suggested in the original work. Additionally, we tested five active learning threshold values between $10^{-5}$ and $10^{-9}$ to obtain a reduced final training set size with good effectiveness in the ALD baseline. In case of ALIAS, we implemented the method using a committee of decision trees, as suggested in the original work. The similarity functions used to produce the features are the same used with T3S and the pairs are produced by using our proposed blocking strategy (Section 4.1). In case of Christen’s work, as suggested by the author, we select 10 percent of the available pairs to produce the training set (without user intervention). As before, we used the same similarity functions and blocking strategy of T3S.

5.4 Identifying the Initial Threshold
In these first experiments, we evaluate the strategy for creating the set of candidate pairs. As in this step our goal is to maximize recall while avoiding an excessive generation of candidate pairs, we focused on the identification of the initial threshold value following Definition 2 (Section 3.1). We run experiments using random samples with 1, 5, and 10 percent of the entire dataset.

Fig. 4 reports the results for the Dirty, DBLPxCiteseer and IMDBxNetFlix datasets. In the Y axis, the Pair size/Sample size (P/S) refers to the rate of candidate pairs created over the number of records in the sample. We adopt such normalization to summarize and better illustrate the number of candidate pairs created by each threshold value. If the P/S value is lower than 1 then the number of candidate pairs is lower than the sample size, as in Definition 2. Since all synthetic datasets present similar behavior (i.e., use the same initial threshold), we detail here just the results for the Dirty dataset.

We can notice that, in these first experiments, the synthetic and real datasets produced different curves regarding the size of the set of candidate pairs. This is somewhat expected due to the data patterns and noise levels in each

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of records</th>
<th>Number of matching pairs</th>
<th>Average length of each record</th>
<th>Number of tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean</td>
<td>105,000</td>
<td>5,000 (5%)</td>
<td>82</td>
<td>235,362</td>
</tr>
<tr>
<td>Dirty-Clean</td>
<td>120,000</td>
<td>20,000 (20%)</td>
<td>81</td>
<td>257,083</td>
</tr>
<tr>
<td>Dirty</td>
<td>150,000</td>
<td>50,000 (50%)</td>
<td>82</td>
<td>291,639</td>
</tr>
<tr>
<td>DBLP</td>
<td>1,995,539</td>
<td>-</td>
<td>169</td>
<td>2,013,244</td>
</tr>
<tr>
<td>Citeseer</td>
<td>811,408</td>
<td>-</td>
<td>129</td>
<td>644,479</td>
</tr>
<tr>
<td>IMDB</td>
<td>1,080,000</td>
<td>-</td>
<td>55</td>
<td>551,583</td>
</tr>
<tr>
<td>NetFlix</td>
<td>160,000</td>
<td>-</td>
<td>76</td>
<td>70,654</td>
</tr>
</tbody>
</table>

TABLE 1
Datasets Statistics
dataset. The Dirty dataset (Fig. 4A) produces a large set of candidate pairs when using a 0.2 threshold value. This happens because a substantial number of frequent tokens (e.g., stop words) are indexed with a threshold value of 0.2. For instance, the 0.2 threshold with sample size of 1 percent produces a P/S value of 30 (about 450,000 pairs from 15,000 records). On the other hand, considering a threshold value of 0.3 reduces substantially the set of candidate pairs with P/S value lower than one. This qualifies the 0.3 threshold to be used as the initial threshold in the synthetic datasets.

To evaluate the recall reached by the initial threshold, we explored the threshold values of [0.2, 0.3, 0.4, and 0.5] in the entire dataset, shown in detail in Table 2(A). Note that the threshold value of 0.2 retrieves almost all true pairs with drawback of 16 times more candidate pairs than the initial threshold (0.3 threshold). The initial threshold prunes out only 2 percent of true pairs, while threshold values of 0.4 and 0.5 remove about 6.9 and 20 percent of the true pairs. Our strategy successfully finds the initial threshold value that maximizes recall with a feasible set of candidate pairs to be further analyzed.

The IMDBxDBLP dataset (Fig. 4B) produces relatively less candidate pairs than the other datasets. This is because Netflix contains almost nine times less records than IMDB, resulting in a relative low number of candidates. Note that the threshold value of 0.3 with sample size of 1 percent produces P/S value lower than one (about 12,400 pairs), qualifying 0.3 threshold value to be used as initial threshold. However, the same threshold (0.3) with a sample size of 5 and 10 percent produces a P/S value of 2.0 and 3.8, respectively. This happens because the 1 percent sample is not big enough to select the frequent tokens. Then, we adopted a default sample size of 5 percent. In such scenario, the threshold value of 0.4 becomes qualified to be used as the initial threshold as its P/S value is lower than one. Table 2(B) shows the recall and the number of candidate pairs produced by each threshold in the entire dataset. The initial threshold recovers all true pairs. The threshold value of 0.3 recovers all true pairs with the drawback of producing more than 24 million of candidates than the set created by the initial threshold.

The DBLPxCiteeseer dataset results (Fig. 4C) show a higher number of candidate pairs compared to the other datasets. For instance, a 0.3 threshold value with a sample of 10 percent produces a P/S value of 90. The large number of candidate pairs in DBLPxCiteeseer dataset is a combination of both, large dataset size and record length (as illustrated in Table 2(C)). Such characteristics force the Sig-Dedup filters to be more aggressive, avoiding the indexing of more frequent tokens. In this context, 0.6 value is defined as initial threshold since it is the first one to produce a P/S value lower than one.

Table 2(C) illustrates the recall behavior as we vary the threshold value in the entire DBLPxCiteeseer dataset. Recall close to the maximum is achieved by the initial threshold with value of 0.6. When the threshold value is changed to 0.7, recall drops by almost 10 percent. The 0.5 threshold value recovers all true pairs with the drawback of almost eight times more candidate pairs than that selected by the initial threshold value.

Overall we claim that the proposed strategy properly identifies the initial threshold value that achieves the maximum (or close to the maximum) recall and a feasible set of candidate pairs, without user intervention.

### 5.5 First Stage Sampling versus Random Selection

In this section we analyze the impact of T3S’s first stage (a sampling selection using fixed levels) in the whole process by substituting it by a random selection in the entire set of candidate pairs (abbreviated as random selection strategy). In other words, we experimentally demonstrate that the T3S first stage is an effective way to produce a balanced and informative set of pairs to be exploited by the T3S second stage (i.e., incremental active learning process). We have only reported the results with SVM as the random selection strategy selects few matching pairs. These are not enough to identify the NGram threshold of T3S-NGram, producing an almost empty output set. Additionally, in this section we only discuss experiments based on the Dirty dataset because the labeled sets for the real datasets had already been selected by using our first stage sampling selection, and results in the other synthetic datasets were very similar.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Th</th>
<th>Recall</th>
<th>#pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) Dirty</td>
<td>0.2</td>
<td>0.990</td>
<td>43,629,641</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.979</td>
<td>2,776,461</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>0.935</td>
<td>263,872</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.833</td>
<td>90,292</td>
</tr>
<tr>
<td>(B) IMDBxNetFlix</td>
<td>0.3</td>
<td>1.00</td>
<td>33,730,220</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>1.00</td>
<td>8,381,906</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.99</td>
<td>2,340,065</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.94</td>
<td>243,109</td>
</tr>
<tr>
<td>(C) DBLPxCiteeseer</td>
<td>0.5</td>
<td>1.00</td>
<td>69,313,296</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.99</td>
<td>13,180,090</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.9</td>
<td>2,122,995</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.7</td>
<td>317,818</td>
</tr>
</tbody>
</table>
Fig. 5 illustrates the comparison. In the figure, the levels sizes (10, 50, 100, 500 and 1,000) specify how many pairs are randomly selected in each similarity level. This corresponds to 100 pairs (10 pairs × 10 levels), 500 pairs, and so on. In the case of the Random selection, we randomly select the same amount of pairs, without considering the levels. We can see that in the beginning of the process, there is some improvements in terms of F1 for both, the random selection and T3S, as the size of the selection increases. This happens because with a larger selection there is a higher chance of selecting more positive pairs, which are necessary to train the classifier.

However, when we increase the size of the selected set further, results for the Random selection start to decay while for T3S they become stable (results are statistically tied). This happens because in the random selection there is a much higher chance of selecting negative pairs as they are dominant in the actual distribution of the repository, i.e., most of the pairs indeed do not correspond to replicas. This causes a natural bias of the classifier towards the negative class, which increases when training with more negative pairs. When the selection considers the similarity levels, as with T3S, this natural bias is smoothed as we have a higher chance of getting more positive pairs from the levels with higher similarities.

Notice that the stop point for the labeling process is naturally chosen by SSAR’s stopping criterion.

### 5.6 T3S versus FS-Dedup

This section examines the results obtained when comparing the labeling effort and effectiveness of the proposed T3S-(SVM and NGram) with FS-Dedup-(SVM and NGram). This comparison is interesting as FS-Dedup uses a random selection of pairs inside the fuzzy region to produce the training set, while T3S uses a random selection in the first stage to produce representative samples combined with an incremental active learning approach that can remove the redundant information.

Tables 3 and 4 provide details of the manual effort and the effectiveness of the deduplication process by comparing T3S-SVM with FS-Dedup-SVM and T3S-NGram with FS-Dedup-NGram. In these tables, the first column specifies the level sizes of 10, 50, 100, 500, and 1,000 pairs. The effectiveness with their standard deviation values (\(\sigma\)) are shown in the “F1” columns. The average number of labeled pairs (without deviations) is reported in the “#Pairs” columns. Finally, the columns “% sel” show the relative reduction in the training set size compared to the labeling effort of T3s with FS-Dedup. We use the same training set for T3S-SVM and T3S-NGram in these experiments and replicate the value of the training set size in Tables 3 and 4 to better illustrate this.

To summarize, Table 3(A) shows that T3S-Ngram reduces the training set size considerably with an equivalent F1 value, using only one similarity function (T3S-Ngram(1SF)) in synthetic datasets. T3S-NGram does not show any improvements in the F1 value when more similarity functions are used (i.e., T3S-Ngram(2SF)). In fact, the training set created by T3S-Ngram(1SF) proves to be enough to provide information that can enable the NGram
threshold to be identified correctly. Naturally, when the level size increases (i.e., to 100, 500, and 1,000 pairs) the reduction in the training set size is more expressive due to the large number of redundant pairs that are randomly selected at each level. Exceptionally, when the smallest level size (i.e., 10 pairs) is employed, T3S-Ngram relies on more labeled pairs: around 1.7 times more than the FS-Dedup-Ngram, though in this scenario none of the approaches achieve reasonable effectiveness. When more pairs are randomly selected at each level (e.g., 100, 500, and 1,000 pairs) the T3S-Ngram is able to prune out a large number of pairs without any loss of effectiveness. For instance, when a level size of 100 pairs is used, T3S-Ngram reduces the training set size by more than 50 percent and basically ties the F1 value. The reduction in the training set size is even more extreme at the level size of 1,000 pairs, and reaches 93 percent (compared with 5,127, 5,002 and 5,902 pairs used by FS-Dedup, T3S-Ngram selects 333, 331, and 346 pairs in the Clean, Dirty-Clean and Dirty datasets, respectively). Clearly, this demonstrates that a large number of pairs used by FS-Dedup-Ngram are composed of redundant information which does not help to configure the classification process and can removed without any impact on effectiveness.

Table 4(A) shows that T3S-SVM(1SF) obtains a statistically equivalent F1 value in all level sizes compared with FS-Dedup-SVM using the synthetic datasets, as does T3S-Ngram. However, in these datasets the level size of 1,000 pairs has an unstable behavior compared with FS-Dedup-SVM. The reason for this is that in contrast with the other level sizes, the level size of 1,000 pairs identifies a very large fuzzy region. Although a large fuzzy region is important to improve the number of matching pairs (i.e., improve the recall), it requires the training set to include most of the dataset patterns so that it can produce an effective classification model. For instance, the T3S-SVM(1SF) that uses a Clean dataset, achieves an unstable F1 value of 84 percent compared with the 92 percent reached by FS-Dedup-SVM with a level size of 1,000 pairs. In other words, the T3S-SVM (1SF) has an unpredictable behavior that, in some cases, can result in lower effectiveness. These results may be explained by the fact that the synthetic datasets are composed of matching pairs with inversion of data fields (e.g., changes in the positions of first and surname values in a citation), which are easily identified by T3S-Ngram (i.e., resort the record using a global frequency). Since the SVM depends on a well formatted input set, the inversion of data fields leads to a depreciation in effectiveness.

T3S-SVM with two similarity functions (T3S-SVM(2SF)), achieves a more stable F1 values of 94, 93, and 96 percent and is able to maintain a removal rate of more than 90 percent of pairs required by FS-Dedup-SVM to produces similar results in the Clean, Dirty-Clean an Dirty datasets, respectively. This illustrates that the use of T3S-SVM with two similarity functions produces a more informative training set that is able to create an effective training model.

Interestingly, both T3S-Ngram and T3S-SVM in the real datasets (IMDBxNetFlix and DBLPxCiteeseer) reach competitive F1 values with only one similarity function (T3S-Ngram(1SF) and T3S-SVM(1SF)) with a substantial reduction in the training set size compared to FS-Dedup, as shown in Tables 3(B) and 4(B). For instance, when a level size of 100 pairs is used, T3S only selects 10 percent (73 pairs selected from a set of 696 pairs) and 13 percent (75 pairs selected from a set of 581 pairs) and basically ties the F1 value if a comparison is made with FS-Dedup-Ngram in the IMDBxNetFlix and DBLPxCiteeseer datasets, respectively. Notice also, the dataset DBLPxCiteeseer achieves an equivalent F1 value using the level size of 10, 50, and 100 pairs. In other words, in this dataset with only 27 manually labeled pairs the T3S-(Ngram-SVM)(1SF) achieves maximum effectiveness (i.e., F1 value of 91 percent) compared with FS-Dedup.

### Table 4

Synthetic and Real Datasets Results Comparing T3S-SVM with FS-Dedup-SVM (Best Results are Highlighted in Bold)

<table>
<thead>
<tr>
<th>Dataset Size</th>
<th>Level</th>
<th>FS-Dedup-SVM</th>
<th>T3S-SVM (1SF)</th>
<th>T3S-SVM (2SF) &amp;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pairs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A) Clean</td>
<td>10</td>
<td>0.87 ± (0.07)</td>
<td>0.81 ± (0.07)</td>
<td>0.81 ± (0.13)</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.89 ± (0.07)</td>
<td>0.85 ± (0.07)</td>
<td>0.87 ± (0.05)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.88 ± (0.10)</td>
<td>0.90 ± (0.01)</td>
<td>0.89 ± (0.03)</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.89 ± (0.04)</td>
<td>0.91 ± (0.06)</td>
<td>0.90 ± (0.07)</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>0.92 ± (0.04)</td>
<td>0.84 ± (0.24)</td>
<td>0.94 ± (0.02)</td>
</tr>
<tr>
<td>Dirty-Clean</td>
<td>10</td>
<td>0.87 ± (0.07)</td>
<td>0.88 ± (0.11)</td>
<td>0.85 ± (0.11)</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.92 ± (0.03)</td>
<td>0.92 ± (0.05)</td>
<td>0.93 ± (0.03)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.92 ± (0.03)</td>
<td>0.92 ± (0.05)</td>
<td>0.93 ± (0.03)</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.92 ± (0.03)</td>
<td>0.88 ± (0.08)</td>
<td>0.95 ± (0.01)</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>0.94 ± (0.02)</td>
<td>0.85 ± (0.26)</td>
<td>0.93 ± (0.05)</td>
</tr>
<tr>
<td>Dirty</td>
<td>10</td>
<td>0.92 ± (0.02)</td>
<td>0.90 ± (0.05)</td>
<td>0.94 ± (0.02)</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.92 ± (0.07)</td>
<td>0.92 ± (0.03)</td>
<td>0.92 ± (0.11)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.94 ± (0.04)</td>
<td>0.94 ± (0.04)</td>
<td>0.93 ± (0.07)</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.96 ± (0.01)</td>
<td>2.752 ± (0.01)</td>
<td>0.95 ± (0.01)</td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>0.94 ± (0.02)</td>
<td>5,902 ± (0.18)</td>
<td>0.96 ± (0.00)</td>
</tr>
<tr>
<td>(B) IMDBxNetFlix</td>
<td>10</td>
<td>0.86 ± (0.01)</td>
<td>0.80 ± (0.03)</td>
<td>0.80 ± (0.05)</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.88 ± (0.01)</td>
<td>0.88 ± (0.06)</td>
<td>0.91 ± (0.04)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.88 ± (0.00)</td>
<td>0.90 ± (0.01)</td>
<td>0.91 ± (0.00)</td>
</tr>
<tr>
<td>DBLPxCiteeseer</td>
<td>10</td>
<td>0.94 ± (0.01)</td>
<td>0.92 ± (0.02)</td>
<td>0.93 ± (0.00)</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.94 ± (0.01)</td>
<td>0.92 ± (0.01)</td>
<td>0.93 ± (0.01)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.94 ± (0.00)</td>
<td>0.94 ± (0.00)</td>
<td>0.93 ± (0.00)</td>
</tr>
</tbody>
</table>
The smaller variations in the effectiveness of DBLPxCiteSeer dataset (less than 2 percent) with different level sizes may be explained by the fact that this dataset is composed of automatically crawled data from the Web. This results in pairs with fewer mistakes, with most of the differences justified by incomplete information, which is more easily coped by the classification approaches, even using fewer labeled pairs to train the classifier.

In summary, the experiments show that T3S-NGram requires much fewer labeled pairs to achieve a stable F1 value in synthetic and real datasets than FS-Dedup-SVM. In the synthetic datasets, T3S-SVM depends on two similarity functions to achieve stable effectiveness. In the real datasets, both T3S-SVM and T3S-NGram (together with one similarity function and a level size of 100 pairs) achieve competitive effectiveness and use only 10 and 13 percent of the training set required by FS-Dedup in IMDBxNetFlix and DBLPxCiteSeer datasets, respectively.

5.7 T3S versus ALIAS, ALD and Christen (2008)

We present experiments with the real-world and one synthetic datasets (results in the other two synthetic datasets are very similar). ALIAS [20] uses a committee of classifiers to actively select informative pairs in order to reduce labeling effort. ALD [4] corresponds to a recent active learning deduplication method and works by looking for highly informative pairs among all unlabeled pairs. Finally, we report experiments with the unsupervised approach proposed by [8], called Christen (2008). It should be noticed that in the case of the real-world datasets, since they are constituted of an preselected smaller and more balanced subset of labeled pairs selected with T3S’s first stage, we are giving an unfair advantage to the baselines. Even this being the case, to illustrate the potential of our approach, we included experiments comparing ALD, ALIAS and Christen (2008) with T3S in the real datasets.

Fig. 6 shows the experimental results. It can be seen, that T3S-[NGram and SVM] converge very quickly, producing good effectiveness with only a few manually labeled pairs (around 380, 103 and 31 pairs in Clean, IMDBxNetFlix and DBLPxCiteSeer datasets, respectively). Note that, T3S clearly outperforms ALIAS with a reduced labeling effort in both real datasets. ALIAS has also an unstable behavior at the beginning, while T3S starts with a high F1 value. In the synthetic dataset ALIAS achieves a competitive F1 and label efforts when compared to T3S because these datasets have a pattern of dirtiness that can be easily identified by the decision tree classifier (i.e., the feature “social security number” is highly informative). Yet, in the very beginning, T3S is superior. In the IMDBxNetFlix and DBLPxCiteSeer datasets, ALIAS depends on 130 and 140 pairs to reach effectiveness levels of 90 and 92 percent, respectively. T3S-SVM requires only 103 and 31 labeled pairs (a reduction of 21 and 78 percent), reaching a statistically significant gain of 3 percent in the F1 value in the IMDBxNetFlix dataset and tying in the DBLPxCiteSeer dataset. Interestingly, ALIAS has a drop in the F1 value when more pairs are labeled, e.g. in the IMDBxNetFlix dataset the effectiveness drops in about 2 percent. This behavior has been observed in the experimentation performed in [2].

The ALD baseline, which always starts with a sample size of around 30 pairs to configure the oracle, initiates with a low effectiveness value in all datasets. The convergence point occurs with around 410, 166, and 161 labeled pairs and F1 values of 87, 88 and 75 percent in the Clean, IMDBxNetFlix and DBLPxCiteSeer datasets, respectively. In the Clean and DBLPxCiteSeer datasets, T3S converges with a reduction of labeling effort of around 26 and 79 percent, compared with ALD, tied in terms of F1. In IMDBxNetFlix, T3S can achieve a labeling effort reduction of 87 percent (765 pairs are selected by ALD compared with 103 selected by T3S), and still be around 6 percent better than ALD in terms of F1. In both Clean and DBLPxCiteSeer, ALD reaches a stable F1 value of 90 percent with only a very large training set (around 2,000 and 800 pairs, respectively). In sum, a substantially higher number of pairs must be labeled using ALD to achieve a competitive effectiveness when compared to T3S.

Christen (2008) achieves very poor effectiveness: 21, 68 and 70 percent in the Clean, IMDBxNetFlix and DBLPxCiteSeer datasets, respectively. This matching quality is a result of the low informativeness of the automatically produced training set which misses true matching pairs belonging to the fuzzy region and due to the propagation of label mistakes when very dissimilar pairs are wrongly labeled. Very poor results with the Clean are specially due to very dissimilar pairs considered as a matching (e.g. a pair with similarity degree of 0.11 may represent a matching). This produces a propagation of labeling mistakes in the automatic training set generation. This demonstrates that the deduplication task indeed requires some manual tuning to achieve a high quality output.

In summary, compared with the baselines, T3S sharply reduces the training set size and, consequently, the manual labeling effort, while producing competitive or superior results in terms of effectiveness when compared with the state-of-the-art deduplication methods.

5.8 Efficiency

In DBLPxCiteSeer, with about one trillion of potential comparisons, T3S takes around 74 minutes in a single machine.
6 CONCLUSIONS

We have proposed T3S, a two-stage sampling strategy aimed at reducing the user labeling effort in large scale deduplication tasks. In the first stage, T3S selects small random subsamples of candidate pairs in different fractions of datasets. In the second, subsamples are incrementally analyzed to remove redundancy. We evaluated T3S with synthetic and real datasets and empirically showed that, in comparison with four baselines, T3S is able to considerably reduce user effort while keeping the same or a better effectiveness. For future work, we intend to investigate genetic programming to combine similarity functions and investigate whether it is possible to provide theoretical boundaries on how close our MTP and MFP boundary estimates are to the ideal values.

ACKNOWLEDGMENTS

Most of this work was done while Guilherme Dal Bianco was at the Instituto de Informática, UFRGS. G. D. Bianco is the corresponding author.

REFERENCES


Guilherme Dal Bianco is an assistant professor at the SENAI College of Technology, Porto Alegre, Brazil.

Renata Galante is a associate professor in the Computer Science Department of the Federal University of Rio Grande do Sul (UFRGS).

Marcos André Gonçalves is an associate professor of computer science at the Universidade Federal de Minas Gerais.
Sergio Canuto is working towards the PhD degree in computer science at the Federal University of Minas Gerais.

Carlos A. Heuser is a full professor at the Universidade Federal do Rio Grande do Sul, Brazil.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.
Query to the Author

Q1. Please provide report number and location of institution for Ref. [10].