Automatic Selection of Training Examples for a Record Deduplication Method Based on Genetic Programming

Gabriel S. Gonçalves, Moisés G. de Carvalho, Alberto H. F. Laender, Marcos A. Gonçalves

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
31270-901 – Belo Horizonte – MG, Brasil
{gabriels, moisesgc, laender, mgoncalv}@dcc.ufmg.br

Abstract. Recently, machine learning techniques have been used to solve the record deduplication problem. However, these techniques require examples, manually generated in most cases, for training purposes. This uneases the use of such techniques because of the cost required to create the set of examples. In this article, we propose an approach based on a deterministic technique to automatically suggest training examples for a deduplication method based on genetic programming. Our experiments with synthetic datasets show that, by using only 15% of the examples suggested by our approach, it is possible to achieve results in terms of F1 that are equivalent to those obtained when using all the examples, leading to savings in training time of up to 85%.

Categories and Subject Descriptors: H. Information Systems [**H.3. Information Storage and Retrieval**]: Digital Libraries; I. Computing Methodologies [**I.2. Artificial Intelligence**]: Problem Solving, Control Methods, and Search

Keywords: Replica Identification, Artificial Intelligence, Genetic Programming

1. INTRODUCTION

The volume of data generated and stored by companies and organizations has been increasing significantly. According to a report released by Enterprise Strategy Group[1], a known market analysis company, 13% of the media companies analysed in 2004 stored and used more than 10 terabytes of data. In 2008, this number had increased to 42%, i.e., more than three times in a time period shorter than four years [Geer 2008].

Thus, due to this increasing in the volume of stored data, administrators of large data repositories, such as digital libraries and large corporate databases, have been facing constant problems to maintain the quality of the available data in their repositories. Since many of these repositories are built and complemented by integrating several data sources, it is possible that several inconsistencies exist, allowing the generation of duplicated data, thus resulting in repositories with "dirty data".

Currently, it is possible to establish a relationship between the quality of the data available in the repositories of an organization and its capability of delivering high quality services to its clients. The decision of keeping repositories with "dirty data" goes beyond technical issues, affecting the

[1] http://www.enterprisestrategygroup.com/

This work is partially funded by the National Institute of Science and Technology for the Web – INCT Web (grant number MCT/CNPq 550874/2007-0), by the InfoWeb project (grant number MCT/CNPq/CT-INFO 573871/2008-6), and by the authors' individual scholarhips and grants from CNPq and FAPEMIG.

performance and quality of the systems that use these data. Besides technical efforts, cultural and management changes are also necessary to help with such problems [Bell and Dravis 2006].

The maintenance of repositories with "dirty data" can cause several problems. For example, the performance of a database management system can be seriously affected since useless data demand additional processing, requiring more time to answer even simple queries. The quality of the information that can be extracted from the repositories is also jeopardized given that duplicates and inconsistencies may generate distortions in reports, causing the making of wrong decisions. Besides, an increase in operational costs is also expected since an (unnecessary) elevation in the data volume may incur in higher investments in storage and computational processing capacity with the goal of maintaining response time in acceptable levels.

The problem of detecting and removing duplicates in data repositories is known as *record deduplication* [Koudas et al. 2006], but it is also denominated in the literature *data cleaning* [Chaudhuri et al. 2003], *record linkage* [Bhattacharya and Getoor 2004; Fellegi and Sunter 1969], and *record matching* [Verykios et al. 2003]. More specifically, record deduplication consists of identifying and removing, from data repositories, records referring to the same object or entity in the real world, even if writing styles, data types or schemas are differemt.

Lately, there has been a lot of investment by companies and governmental institutions in the development of effective methods for removing duplicates in large data repositories [Bell and Dravis 2006; Wheatley 2004]. However, record deduplication is a complex task whose treatment requires a lot of time and processing power due to the large amount of record comparisons necessary. Thus methods proposed to deal with record deduplication should try to achieve their goal as efficiently as possible.

Recently, de Carvalho et al. [2008] have proposed an innovative method for record deduplication that is based on a machine learning technique known as *Genetic Programming (GP)* [Banzhaf et al. 1998; Koza 1992]. Through this method, records are deduplicated using evidence extracted from the data content to create similarity functions, generically called *deduplication functions*, capable of pointing out which records of a repository are replicas. However, despite superior results when compared to other approaches found in the literature, machine learning techniques generally rely on a training phase in which examples for learning duplication patterns are usually generated manually. As such, the cost and time necessary for creating the training sets make it difficult to use such techniques in practice.

In this article, we propose an approach based on a deterministic technique that automatically suggests examples for the training phase of de Carvalho et al.'s GP-based record deduplication method. Initially, we verify the real need of using all the training examples generated for the training phase. For this, we performed several experiments in which the examples of duplicated pairs of records were gradually reduced in order to verify how each reduction affected the effectiveness and performance of the process of generating deduplication functions. Next, a deterministic method was used to generate training examples for the deduplication process using GP, allowing an analysis of the viability of automatically selecting these examples. Our experimental results show that it is possible to use a reduced set of training examples without affecting the quality of the obtained solutions in the end of the process of generating deduplication functions, significantly reducing the time necessary for the execution of the training phase.

This article is organized as follows. In Section 2, we discuss related work. In Section 3, we present a general view of the GP-based record deduplication method. In Section 4, we describe how our approach for the automatic selection of training examples works. In Section 5, we present our experiments and discuss the obtained results. Finally, in Section 6 we have our final considerations and discuss possible future work.

2. RELATED WORK

Record deduplication is a research topic that has attracted a lot of attention in the database field and related areas. As already mentioned, the existence of replicas in data repositories can lead to inconsistencies that may severely affect several types of service, causing damage for companies and governmental institutions.

In an attempt to solve these inconsistencies, some works have proposed the creation of similarity functions capable of combining information contained in the data repositoryies in order to identify when a pair of records constitutes or not a replica. Elmagarmid et al. [2007] classify these approaches into two categories, exemplified next:

Ad-Hoc. These are methods that usually depend on some knowledge of a particular domain or on specific distance metrics (for example, for strings characters)

Training-based. These are methods that depend on some type of training, supervised or semi-supervised, for the identification of replicas, such as probabilistic and machine learning approaches.

In the following, we briefly describe some rellevant work that illustrates these two categories of methods.

2.1 *Ad-hoc* Methods

Chaudhuri et al. [2003] have proposed a linkage algorithm that receives a record from an archive or data repository as input and searches for another record in a reference archive that "matches" with the former, according to some predefined similarity function. The matched records are selected according to some user defined minimum similarity threshold, allowing that more than one candidate record be returned as a possible answer. In this case, the user is responsible for choosing the duplicated record most similar to the original one.

An information retrieval style method is used in WHIRL [Cohen 2000], a database management system that supports similarity joins between relations that include textual attributes. In this system, textual attributes are represented according to the *vector space model* [Salton 1989] and term weights are calculated using the well-known TF-IDF scheme [Baeza-Yates and Ribeiro-Neto 1999] in order to determine when tuples from two relations should be joined based on the similarity of their attribute values.

Carvalho and da Silva [2003] also use the vector space model for computing similarity among fields from diferrent data sources and evaluate four distinct strategies to assign weights and combine the similarity scores of each field. As a result of their experiments, they found that using evidence extracted from individual attributes improves the result of the replica identification task.

2.2 Training-based Methods

Since these are more related to the work presented in this article, we cover training-based methods in more details in this section.

Newcombe et al. [1959] were the first ones to address the record deduplication problem as a Bayesian inference problem and proposed the first method to automatically handle replicas. However, their method was considered empirical [Elmagarmid et al. 2007] since it lacks a deeper statistical ground.

After Newcombe et al.'s work, Fellegi and Sunter [1969] have proposed an elaborated probabilistic method to deal with the problem of evidence combination. Their proposed method requires the definition of two thresholds for replica identification. If the similarity value between two records is above the *positive threshold*, these records are considered as replicas; if it is below the *negative*

threshold, the records are considered non-replicas; and if the similarity value is between these two thresholds, the records are classified as "possible replicas", requiring a manual classification by a specialist.

The Fellegi and Sunter's method does not require explicit training examples but, on the other hand, it does require that the data administrator explicitly defines the two aforementioned thresholds, a non trivial task since this definition is repository-dependent. This method has dominated the field for more than two decades until new deduplication methods were developed by the statistical and machine learning communities. Febrl[2] [Christen 2008] is one of the best known software packages that implement this method.

On the other hand, pure machine learning based methods need some training examples, which should present characteristics similar to those of the duplicated data, so that they can generalize their solutions (deduplication functions) for the entire repository. The main problem with this type of method is the cost of creating such training data, which in some cases may be infeasible.

Bilenko and colleagues [Bilenko et al. 2003; Bilenko and Mooney 2003] exploit a machine learning technique to improve both the similarity functions that are used to compare record fields and the way pieces of evidence are combined to identify replicas. In their work, extracted evidence is encoded as feature vectors, which are then used to train an SVM (*Support Vector Machines* [Joachims 2002]) classifier to better combine them for the replica identification task that is performed by a system called MARLIN (*Multiply Adaptive Record Linkage with INduction*). This system also uses several blocking strategies to improve its effectiveness when clustering similar records.

Active Atlas [Tejada et al. 2001] is an object identification system that aims at learning mapping rules for identifying similar objects (records) from distinct data sources. The process involves two steps. First, a candidate mapping generator proposes a set of possible mappings between the two sets of objects by comparing their attribute values and computing similarity scores for the proposed mappings. Then, a mapping rule learner determines which of the proposed mappings are correct by learning the appropriate mapping rules for that specific application domain. This learning step is exectued by using a decison tree. What mainly differs the method behind this system from other machine learning based methods is that it tries to reduce the training effort by relying on user-provided information for selecting the most relevant training examples.

In [de Carvalho et al. 2006], the authors follow a genetic programming approach to improve the results of the Fellegi and Sunter's method. They apply this machine learning technique to balance the weight vectors produced by the Fellegi and Sunter's method in order to generate a better evidence combination than the simple linear summation used by that probabilistic model [Fellegi and Sunter 1969].

Following their previous work, de Carvalho et al. [2008] have proposed a new GP-based method that finds the best evidence combination for generating a deduplication function within a generic framework that is independent of any other technique. Since record deduplication is a very costly task, even for small repositories, the proposed method tries to find the best evidence combination that also maximizes performance, using for this only a subset of the repository as training data.

In this article, we present results of an experimental study that shows how the size of the training set required by de Carvalho et al.'s GP-based method affects the quality of its obtained solutions in the end of the evolutionary process. Moreover, a deterministic process is proposed for generating training sets for the record deduplication process, allowing us to analyze the viability of automatically generating training samples without manual intervention.

[2]Freely Extensible Biomedical Record Linkage – http://sourceforge.net/projects/febrl

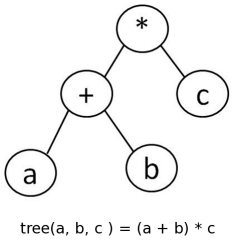

Fig. 1. Example of a Function mapped as a Tree.

3. OVERVIEW OF THE GP-BASED RECORD DEDUPLICATION METHOD

In this section we provide an overview of the GP-based record deduplication method used in this work and first introduced in [de Carvalho et al. 2008]. We start by presenting a brief introduction to the genetic programming technique followed by an explanation of how the record deduplication problem was modeled using this technique. We conclude with a more detailed description of the main steps of the method.

3.1 Genetic Programming Overview

Genetic Programming (GP) is one of the best known and mostly used techniques of evolutionary computing and can be seen as an adaptive heuristics whose basic ideas are inspired in the natural selection process. It is a direct evolution of programs or algorithms used for inductive (supervised) learning, initially applied to optimization problems.

One of the main characteristics of evolutionary techniques is their capability to deal with problems with multiple objectives, normally modeled as environmental constraints during the evolutionary process [Banzhaf et al. 1998]. These techniques are also known for their capability of searching for solutions in large, possibly unbounded, search spaces in which the optimal solution may be unknown, thus producing solutions close to the optimum [Banzhaf et al. 1998; Koza 1992].

What mainly differentiates GP from other evolutionary techniques (such as genetic algorithms and evolutionary systems) is the representation of concepts and the solution as a computer program, being the data manipulated only by these programs. Computer programs have the flexibility needed to deal with a multitude of problems. Moreover, the program structures being evolved do not have size limitations and can dynamically vary during the process according to the requirements of the problem [Koza 1992].

The most used GP representation for solutions of a specific problem (also known as an *individual* in a *population*) are trees and graphs. An example of a function that represents a solution for a specific problem that is represented as a tree is depicted in Fig. 1. Besides choosing a representation for the solution of the problem, it is also necessary to define a set of terminals and functions to perform the task of solving the problem. Terminals are inputs, constants, or nodes with no input, which are the leaves of the trees, while the set of functions comprises operators, declarations, and basic or user-defined functions used to manipulate the values of the terminals [Koza 1992]. The leaf nodes are located at the end of the branch while functions are located in the internal nodes, as can be seen in Fig. 1. The search space is the space built from all possible programs that can be contructed with the functions and terminals defined for the problem domain.

A GP algorithm evolves a population of tree-represented individuals, i.e., a set of possible solutions for the problem. In each generation of the evolutionary process, individuals are evaluated according to a specific quality metric that measures how well they solve the problem, calculated by a *fitness* function. The fitness value is used as a criterion to select the best individuals, which will transmit

their characteristics to the future generations through operations like *crossover* and *mutation*. In this work, we use the *ranking method* [Koza 1992] as a selection method. This method chooses the k best individuals based on the fitness values to be included in the next generation and to participate in the genetic operations.

While the number of new individuals is smaller than the desired population size n (a parameter to be set), two individuals are picked among those selected as described above and have their "genetic material" exchanged in the *crossover* operation with a certain probability (another parameter to be set) to generate a new individual that combines the genetic material of their "parents". More specifically, the crossover operation is implemented by randomly choosing one node of each of the two trees (the two individuals) and exchanging the subtrees below them. The role of the crossover operation is to combine good solutions towards the most promising solutions in the search space.

Finally, also with a certain probability (yet another parameter), the *mutation* operation is applied, to introduce new individuals (i.e., solutions) in the population, increasing its diversity. This is useful, for example, to avoid that the whole evolutionary process gets trapped in local minima during the search process for a good solution. In this work, the mutation of an individual (i.e., a tree) is performed by first randomly selecting one of its nodes. We then replace the node (and its corresponding subtree) by a new randomly generated subtree, without exceeding a maximum tree depth d.

The whole process is repeated until a target fitness value f or a maximum number of generations g is reached. In the end of the process, the best individual, i.e., the one with the best fitness value, which usually belongs to the last generation in the evolutionary process, is chosen as the final solution for the problem in hand.

3.2 GP Applied to the Record Deduplication Problem

In this work, as in [de Carvalho et al. 2008], we use a representation based on trees for the individuals of the GP process which, in our case, represent possible record deduplication functions. More specifically, to perform the record deduplication, we use evidence combination functions in which each evidence E is a pair $<$*attribute, similarity function*$>$ that represents the application of a specific similarity function on the values of a given attribute of the data repository. For example, to deduplicate a table from a relational database with attributes *name*, *surname*, *age* and *address*, using the Jaro-Winkler (JW) [Winkler 1999] similarity function, we would have the following pieces of evidence: $E_1<$*name*, JW$>$, $E_2<$*surname*, JW$>$, $E_3<$*age*, JW$>$ and $E_4<$*address*, JW$>$.

For this example, a simple function (F_s) could be a linear combination such as

$$F_s(E_1, E_2, E_3, E_4) = E_1 + E_2 + E_3 + E_4, \tag{1}$$

while a more complex function (F_c) could be

$$F_c(E_1, E_2, E_3, E_4) = E_1 \times (\frac{E_2}{E_3^{E_4}}). \tag{2}$$

To model functions as trees, each evidence is represented as a leaf, by real values normalized between 0.0 and 1.0, while internal nodes represent arithmetic operations (e.g., $+$, $-$, $\times$, $\div$, exp) that manipulate the values in the leaves.

As explained in the previous section, during the evolutionary process, the individuals are manipulated and modified through several genetic operations in a process that tries to generate the best individuals (solutions) in each subsequent generation. All trees generated during this process are automatically evaluated, i.e., each possible solution for the problem is tested in a data repository – with characteristics similar to the training set – where replicas have already been identified. This allows for the automatic evaluation of the capability in identifying pairs of replicas that are true duplicates in large scale problems.

The input to the functions are instances of evidence extracted from the data repository. The output consists of the result of the operation codified in each tree, a value that is compared with a replica identification threshold as follows: if the value is greater than the threshold the records are identified as replicas, otherwise the records are considered as different. This approach obeys the transitivity property of replicas in the sense that, if record A is a replica of record B and B is a replica of C, then A should be a replica of C.

Experiments performed in [de Carvalho et al. 2008] show that the GP-based record deduplication method is able to adapt the suggested deduplication functions according to changes in the replica identification thresholds necessary to classify pairs of records. Therefore, the user does not need to worry about setting up the "best" values for this threshold according to the data repository, given that the suggested deduplication functions can adapt automatically, maintaining the effectiveness of the solutions, despite changes in this threshold.

After a comparison between all pairs of records, there is an assessment of the total number of correctly and incorrectly identified pairs. This information is used a posteriori by the fitness function, the component responsible for evaluating the generated individuals during the evolutionary process. The *F1* metric was chosen as fitness function for the experiments in this work. It combines the traditional metrics of *precision* and *recall* used for evaluation of information retrieval systems [Baeza-Yates and Ribeiro-Neto 1999] as follows:

$$Precision = \frac{NumberOfCorrectlyIdentifiedDuplicatedPairs}{NumberOfIdentifiedDuplicatedPairs} \tag{3}$$

$$Recall = \frac{NumberOfCorrectlyIdentifiedDuplicatedPairs}{NumberOfTrueDuplicatedPairs} \tag{4}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

Precision is responsible for measuring the proportion of correctly classified replicas among all identifications, i.e., of all pairs classified as replicas how many really are. Recall calculates the proportion of correctly identified pairs among all actual pairs existing in the sample, i.e., among all actual pairs how many were identified. Since precision and recall are related metrics, capable of capturing different aspects of the record deduplication process, we decided to use F1 as a single metric because it combines precision and recall. Thus, F1 allows us to have a general view of how good a deduplication function is through a single value (between 0 and 1). High values of F1 imply also in high values of both, precision and recall.

3.3 Record Deduplication using the GP-based Method

The GP-based method for record deduplication proposed by de Carvalho et al. [2008] involves six steps. A detailed description of these steps is given below.

Step 1. A portion of the data repository to be deduplicated is randomly selected for the training phase. In the experiments run in [de Carvalho et al. 2008], the training set corresponded to 25% of the original data repository, given that in the beginning of the deduplication process, the repository was divided into four files, being one of them used for training and the others for evaluation (test) of the generated individuals (solutions).

Step 2. The attributes of the data repository are analyzed and the inputs for the GP method are defined, i.e., we verify the data type of each attribute in order to select a proper similarity function to

each one of them. This is necessary for the creation of the set of evidence used in the representation of the solution as described before. For the experiments performed in this work, the set of evidence was created using the Jaro similarity function for textual attributes and the Edit Distance function for numerical ones. According to experiments performed in [de Carvalho et al. 2008], these similarity functions have been demonstrated to be the most adequate for the respective data types, besides requiring less effort for evidence processing.

Step 3. In the evolutionary process of the proposed GP-based method, individuals (possible solutions for the problem) are generated with the goal of identifying replicas in a given data repository.

Step 4. Within the record deduplication framework, the generated individuals are used to identify replicas in the portion of the repository used for training, generating a report in the end of the evolutionary process. In this report, we have a list of the best individuals in each generation and the respective fitness values, in our case corresponding to the F1 measure.

Step 5. After an analysis of the report on the deduplication process, we select to the next step the best obtained individual, i.e., the function that best performed the deduplication task in the training set, which is usually found in the last generations of the evolutionary problem.

Step 6. The best obtained individual is then used to deduplicate the rest of the repository as well as to deduplicate other repositories with similar characteristics. Notice that new additions of records to the test repository do not prevent the use of the discovered functions as long as the patterns discovered in the data repository do not change much, what is not expected in large repositories.

A clear disadvantage of the described process can be seen in the first step. According to [de Carvalho et al. 2008], the training set is composed of 25% of the original data repository, which corresponds to an unreasonable amount in some scenarios, due to both labeling and training costs. In the following, we present an approach for the automatic selection of training examples. Through a series of experiments we show that it is possible to automatically select a very reduced amount of examples for the training phase while keeping satisfactory levels of effectiveness in the deduplication process.

4. AN APPROACH FOR THE AUTOMATIC SELECTION OF TRAINING EXAMPLES

In this section, we present our proposed approach for the automatic selection of training examples for the GP-based record deduplication method that we consider in this work. Some relevant concepts necessary for the understanding of the process are covered first, followed by a detailed explanation about the functioning of the proposed approach.

4.1 Positive and Negative Pairs of Records

In the beginning of the record deduplication process with genetic programming, the records are grouped in pairs, so they can be compared by pointing out which records in the repository are really replicas. Thus, in order to talk about training examples, it is first necessary to introduce the concepts of *a positive pair of records* and *a negative pair of records*.

A positive pair of records is formed by two records that make reference to the same real-world object or entity, i.e., after comparing the two records against each other using a predefined deduplication function, they are pointed out as replicas. On the other hand, a negative pair of records is constituted of two records that do not make reference to the same real-world object, then being not pointed out as replicas of each other. These two types of training example are then required to help the GP-based record deduplication method to better "understand" what is and what is not a replica within the data repository.

In order to identify replicas in a data repository, each record should be compared to all others. Thus, if a repository contains n records, $\frac{n \times (n-1)}{2}$ comparisons between records should be performed. Notice that only half of the possible comparisons need to be performed, since two records do not need

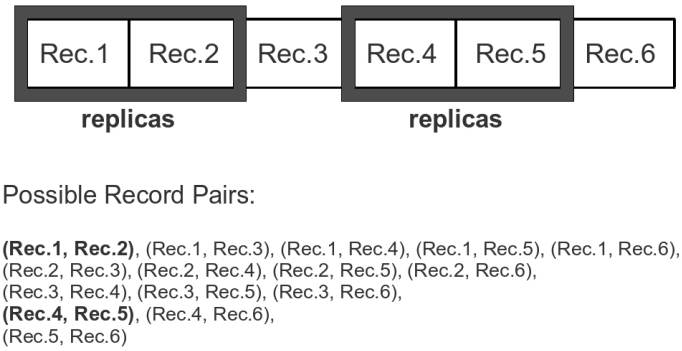

Fig. 2. Example of Positive and Negative Pairs of Records in a Data Repository with six Records (positive pairs are in bold).

to be compared more than once. Most of the comparisons will correspond to non-replicas (negative pairs of records) since the maximum number of duplicated records in a repository is usually much smaller than the amount of records within it [Christen 2008]. Fig. 2 exemplifies positive and negative pairs of records in a data repository composed of six records.

4.2 Automatic Selection of Examples

In [de Carvalho et al. 2008], the generation of deduplication functions requires that the user selects a portion of the repository for the training phase and manually identifies each pair of records as positive or negative. In small data repositories, such task may be realized without much problems depending on the amount of noise and the level of knowledge of the user about the data. However, when large and complex repositories are being deduplicated such manual labeling of training examples may be too costly or even unfeasible.

Another problem regarding the manual generation of training examples is the size of the training set. Defining the amount of pairs (positive and negative) that allows for the generation of good deduplication functions, i.e., that are able to identify the majority of the existing replicas in the data repository as fast as possible, is by no way a trivial task.

To avoid that the user may have to manually create the training set for the generation of the deduplication functions, our proposed approach uses a deterministic technique to automatically select a subset of these examples for the training phase. The steps involved to apply such an approach are the following:

Step 1. The data repository to be deduplicated is divided equally in four files, being one of them for training and the remainder for testing. The experiments presented in Section 5 show the relation between the amount of pairs of records used for training and the results obtained in the end of the process of generation of record deduplication functions with regard to the time spent in the training phase and the quality (effectiveness) of the generated individuals.

Step 2. The deterministic method of Fellegi and Sunter [1969], mentioned in Section 2, deduplicates the training set and generates two lists: one with all positive pairs of records and the other with the negative ones. This deduplication is performed using the Febrl tool [Christen 2008]. Both thresholds are defined always with the same values[3]. This simplification avoids the need to manually identify pairs of records that would be placed between the two thresholds (pairs whose identification is more difficult). To perform the experiments in this work, like in [de Carvalho et al. 2008], the set of

[3]Values defined after initial experiments.

evidence was created using the Jaro similarity function [Koudas et al. 2006] for textual attributes and the Edit Distance function [Koudas et al. 2006] for numerical ones, since these functions, as shown in [de Carvalho et al. 2008], are the most adequate for these data types.

Step 3. Here we define the percentage values of positive and negative pairs of records that will be used in the training phase. The defined amounts of pairs of records is then extracted from the two lists of examples, generated in the previous step, resulting in a new set of training examples.

Step 4. In the end of each generation of the evolutionary process, we select individuals (possible solutions for the problem) with the goal of identifying replicas in an (unseen) data repository.

Step 5. As soon as the evolutionary process ends, a report is generated with the indication of the best individuals in each generation and their respective fitness values, measured by the F1 metric.

Step 6. After analyzing the report generated in the previous step, the user selects, for the next step, the best obtained individual, i.e., the function that performed best in deduplicating the training set.

Step 7. Finally, the best obtained individual is then used to deduplicate the rest of the repository or other repositories with similar characteristics.

As such, the creation of the training set becomes easier, making feasible the practical application of the GP-based method to deal with the problem of record deduplication. This happens because the proposed approach reduces the need of human intervention in the process of creating training examples. It is important to stress that, despite we have used in Step 2 of the proposed approach the method of Fellegi and Sunter [1969], it is possible to use other deterministic classification methods such as k-means [Gu and Baxter 2006].

5. EXPERIMENTS

In this section, we present the results of the experiments performed to demonstrate how the proposed approach for the automatic selection of training examples affects the quality (effectiveness) of the generated solutions as well as the time spent in the training phase of the GP-based record deduplication method.

In the first part of this experimental study, we performed three sets of experiments, gradually varying the percentage of the number of pairs of records (positive and negative) used in the training phase. First, we reduced the number of positive pairs while the number of negative pairs was kept unchanged. Next, the same reduction was applied to the negative pairs, keeping the number of positive pairs in its totality. Finally, we reduced both sets of pairs simultaneously. The values for the percentage of reduction were empirically chosen after initial experiments.

The goal of this first set of experiment is to demonstrate how the choice of the number of examples (positive and negative pairs of records) used in the training phase affects the performance of the GP-based record deduplication method. Moreover, we also wanted to investigate if it is really necessary to use all the generated training examples as was done in [de Carvalho et al. 2008]. The results of this evaluation would allow the suggestion of configurations for the selection of training examples, making it possible the identification of replicas in a more efficient way, without sacrificing the quality of the generated solutions.

In the second part of the experimental study, we used the proposed approach for the selection of training examples to perform a new set of experiments. In this case, we wanted to evaluate, using the F1 metric, if the examples automatically selected by the deterministic method were in fact good ones. This automatic selection makes any machine-learning based record deduplication process more accessible and practical.

As the result of the experiments, we present the mean F1 values and the corresponding standard deviations of the best individual in the test files, after ten executions. The configuration of the GP

Table I. Configuration of the GP Parameters.

Parameter	Value
Max number of generations	30
Max number of executions	10
Population	50
Max tree depth	5
Max mutation tree depth	4
Mutation rate	2%
Initial random population method	Ramped Half-and-Half
Individual selection method	Ranking
Individual pairing method	Random

parameters is shown in Table I. The set of evidence (pairs <attribute, similarity function>) used in this work are the same used in [de Carvalho et al. 2008].

All experiments were performed using workstations with the following configuration: 2 GHz Pentium Core 2 Quad processor, with 4 GB RAM DDR2 of main memory, and HD SATA with 320 GB, running with a FreeBSD[4] 7.1 64-Bits operational system and using version 2.5.1 of the Python[5] programming language.

5.1 Experimental Dataset

For the creation of the datasets necessary to perform our experiments, we used SDG [Christen 2005], a synthetic data generator available in the Ferbl tool [Christen 2008]. This generator allows the creation of datasets containing names (based on frequency tables of names and surnames), addresses (based on frequency tables of locations, postal codes, street numbers, etc.), telephone numbers and personal identifier numbers (social security number). Since real data is not easily available for experimentation mainly due to privacy and confidentiality constraints, the use of synthetic data was considered the best option to test various scenarios. It allows a a better evaluation of the impact in the quality of the final solutions as a result of changes in the number of positive and negative pairs of records used in the training phase, since the characteristics of the existing errors and of the records are completely known.

The data generated by SDG are similar to those commonly found in personal medical data records. First, the tool creates a dataset containing only original records. Next, replicas are generated from these original records by means of small modifications such as removal, insertion and modifications of characters, as well as the removal, insertion and modifications of entire words, changes that are based on characteristics of real errors. Replicas are then inserted in the original dataset to be used in the record deduplication experiments. Each record has the following fields (attributes): *name*, *surname*, *street number*, *address1*, *address2*, *district*, *postal code*, *state*, *birthdate*, *age*, *telephone number* and *social security number*.

For experimentation, a synthetic data repository was created containing 2000 records, distributed equally in four files, each one of them composed of 300 original records and 200 replicas. These replicas were generated obeying the following constraints: at most one replica can be created from a single original record (using a uniform distribution), at most one modification can be made in each attribute of a record and at most one attribute can be modified in the whole record. Since this training file contains 500 records, 124,750 pairs of records will be generated (see discussion in Section 4.1 about this calculation), being 200 positive pairs of records and 124,550 negative pairs.

In the experiments presented in the next section, we used a larger proportion of replicated pairs in the training and test sets than what is normally found in real scenarios, according to the USIIS

[4]http://www.freebsd.org/
[5]http://www.python.org

Table II. Reduction in the Percentage of Positive Pairs of Records – Training Time and Standard Deviation of F1 Values.

Negative Pairs (%)	Positive Pairs (%)	Training Time (hs)	(A) Average	(A) Stand. Deviation	(B) Average	(B) Stand. Deviation	(C) Average	(C) Stand. Deviation
100	100	37.97	0.994	0.009	0.997	0.008	0.995	0.011
	95	41.00	0.996	0.004	1.000	0.000	0.997	0.003
	90	43.37	0.997	0.003	1.000	0.000	0.997	0003
	70	36.80	0.996	0.006	1.000	0.000	0.998	0.001
	50	41.20	0.996	0.002	1.000	0.000	1.000	0.000
	30	40.90	0.995	0.003	1.000	0.000	0.999	0.001
	15	41.42	0.994	0.004	0.998	0.004	0.998	0.004
	5	42.52	0.988	0.010	0.991	0.010	0.989	0.014
	2.5	33.00	0.967	0.035	0.972	0.032	0.967	0.040
	1	32.32	0.952	0.052	0.957	0.049	0.949	0.055
	0	39.62	0.315	0.418	0.317	0.414	0.321	0.419

project[6], that reports that the rate of replicated records is approximately 20%. Despite this, only representative examples, i.e., those most useful for learning duplication patterns, are in fact used for training.

5.2 The Experiments

As explained earlier, the goal of the experiments described in this section is to analyze how the quality of the generated individuals in the deduplication process and the time spent on training are affected by changes in the number of examples used in the training phase of the GP-based record deduplication method. In all experiments, pairs of records were randomly chosen according to the proportions defined in each experiment.

5.2.1 *Experiments with Reduction in the Percentage of Positive Pairs of Records.* In this set of experiments, we gradually reduced the number of positive pairs of records used for training while keeping unchanged the number of negative. Although minority, the positive pairs are determinant for the quality of the generated individuals in the deduplication process. The results of this set of experiments are presented in Table II.

The total time spent – in hours – in the training phase of each experiment of this set is also presented in Table II being useful for comparison purposes. The results obtained in the test phase are presented from the fourth column onwards, indicating the mean value of F1 and the standard deviation of the best individual in each test file (A, B and C). The results of the other sets of experiments are presented in a similar way.

The results show that the reduction in the number of positive pairs of records used in the training phase affects the quality (as measured by the F1 metric) of the results obtained in the test phase. However, when we used very reduced percentages of positive pairs (e.g., 5% and 2.5%) it is still possible to obtain results close to those obtained when all positive pairs are used as examples with high values of mean F1 and low standard deviations, showing that there is a low dispersion of the F1 values with regard to the mean. Also when we use only 2.5% of the positive pairs of records, for example, there is a saving of about 13% in training time. However, it was not possible to obtain a direct relation between the percentage of positive pairs and the time spent for training.

When all positive pairs of records are completely discarded there is a drastic reduction in the F1 values and an increase in the standard deviations. In this situation, the generated individuals identify basically all pairs as replicas, i.e., they are able to identify the most (or even all) positive pairs, but make many mistakes by considering a lot of negative pairs as positive. As such, recall values are close to 1.0 but precision are close to 0.0. Since F1 is a combination of both metrics, the mean F1

[6]Utah Statewide Immunization Information System – http://health.utah.gov/phi/brownbag/handouts/2008/USIIS_april.pdf

Table III. Reduction in the Percentage of Negative Pairs of Records – Training Time and Standard Deviation of F1 Values.

Positive Pairs (%)	Negative Pairs (%)	Training Time (hs)	(A) Average	(A) Stand. Deviation	(B) Average	(B) Stand. Deviation	(C) Average	(C) Stand. Deviation
100	100	37.97	0.994	0.009	0.997	0.008	0.995	0.011
	95	44.13	0.998	0.002	0.999	0.003	0.999	0.001
	90	35.92	0.998	0.003	1.000	0.000	0.998	0.003
	70	26.83	0.998	0.002	0.999	0.003	0.998	0.003
	50	21.08	0.997	0.007	0.999	0.002	0.998	0.001
	30	12.02	0.993	0.009	0.999	0.003	0.994	0.011
	15	5.88	0.995	0.009	0.996	0.007	0.996	0.006
	5	2.33	0.926	0.138	0.925	0.132	0.905	0.185
	2.5	1.35	0.846	0.154	0.851	0.137	0.826	0.183
	1	0.60	0.677	0.242	0.649	0.260	0.654	0.266
	0	0.33	0.163	0.213	0.163	0.221	0.143	0.286

of the best individuals will always be very small, given the high rate of false positives (negative pairs erroneously classified).

5.2.2 *Experiments with Reduction in the Percentage of Negative Pairs of Records.* In this set of experiments, we gradually reduced the number of the most frequent type of pair of records in the training set. Because of that, it is important to analyze how this reduction influences the quality of the individuals generated in the training, as well as the time spent in the execution of this phase.

The results, shown in Table III, demonstrate that the reduction in the number of negative pairs of records influence the quality of the obtained results in a more significant way than the reduction of positive pairs.

In Table II, for example, we observe that the mean values of F1 for the configuration defined with 1% of the positive pairs and the totality of the negative pairs are close to 0.950 while the configuration with 5% of negative pairs and the totality of the positive pairs, see Table III, leads to inferior results than the previous configuration. In any case, using certain percentages of reduced negative pairs (for example, 15%) it is possible to obtain solutions close to those using all training examples. In this case, the saving in training time achieves approximately 85%.

Moreover, we observe a direct relation between the number of negative pairs used for training and the total time spent in this phase, i.e., the smaller the number of negative pairs used the faster is the training phase and vice-versa. This reduction occurs in this set of experiments because the negative pairs correspond to the vast majority of the examples used during training.

In this repository, when we reduce 85% of the negative pairs of records, for example, 105,868 pairs are not considered for the training phase, while the same percentual reduction in the positive pairs only removes 170 pairs of records. As such, we are able to obtain a significant saving in training time since the number of pairs of records used in the training phase is much reduced.

5.2.3 *Experiments with Reduction in the Percentage of Positive and Negative Pairs of Records.* Finally, in this set of experiments, we gradually reduced, at the same time and at the same proportion, both, the number of positive and negative pairs of records. The results are presented in Table IV.

Again, we observe a direct relation between the number of used pairs (positive and negative) and the time spent in the training phase. As can be seen, it is possible to significantly reduce the number of used pairs and still obtain good results. Using only 10% of positive and negative pairs, for example, the saving in training time is of approximately 88% with losses in the quality of the deduplication of only 0.6%, 0.7% and 0.3%, in test files A, B, and C, respectively.

Table IV. Reduction in the Percentage of Positive and Negative Pairs of Records – Training Time and Standard Deviation of F1 Values.

Positive Pairs (%)	Negative Pairs (%)	Training Time (hs)	(A) Average	(A) Stand. Deviation	(B) Average	(B) Stand. Deviation	(C) Average	(C) Stand. Deviation
100	100	37.97	0.994	0.009	0.997	0.008	0.995	0.011
50	50	22.03	0.995	0.002	1.000	0.000	0.999	0.001
25	25	10.85	0.994	0.004	0.996	0.006	0.998	0.002
10	10	4.62	0.988	0.012	0.990	0.011	0.992	0.010
5	5	2.38	0.936	0.142	0.942	0.132	0.924	0.189
2.5	2.5	1.10	0.941	0.051	0.948	0.051	0.952	0.042
1	1	0.67	0.869	0.110	0.875	0.118	0.852	0.150

Table V. Using the Fellegi and Sunter Method for the Generation of the Training Set – Training Time and Standard Deviation of F1 Values.

Positive Pairs (%)	Negative Pairs (%)	Training Time (hs)	(A) Average	(A) Stand. Deviation	(B) Average	(B) Stand. Deviation	(C) Average	(C) Stand. Deviation
10	10	2.20	0.985	0.011	0.980	0.021	0.974	0.024
5	5	1.41	0.979	0.017	0.975	0.020	0.974	0.025
2.5	2.5	1.07	0.986	0.014	0.982	0.023	0.978	0.022

5.3 Deterministic Record Deduplication

The goal of this last set of experiments was to validate our proposed process for the automatic selection of training examples. In all performed experiments, the pairs of records suggested by the deterministic method were randomly chosen using the proportions defined for each experiment. After the experiments of the last sections, we learned that it is possible to satisfactorily use smaller sets of training examples with positive and negative pairs. As such, in the experiments performed here we only used configurations with small percentages of training examples. The results can been seen in Table V.

In this set of experiments, using examples automatically suggested by the Fellegi and Sunter method [1969], the individuals generated in the end of the deduplication process presented high values of mean F1, basically eliminating the need for manual labeling of examples. It was possible to use only 2.5% of positive and negative pairs of records in the training phase without incurring in considerable loss in the quality of the generated individual by the end of the deduplication process. The values of mean F1, when compared with the configuration that uses all the available training data, had a loss in performance of only 0.8%, 1.5% and 1.7% for the test files A, B, and C, respectively. The reduction in training time, on the other hand, was of about 97%.

In sum, we can say that it is possible to satisfactorily exploit the proposed approach for the automatic selection of training examples for the GP-based record deduplication method, at least when working with data repositories with similar characteristics as the ones we used in this work.

6. CONCLUSIONS AND FUTURE WORK

The identification and removal of duplicates is very important for maintaining the quality of the information available in current data repositories. Systems that depend on data integrity in order to offer high quality services, such as digital libraries and e-commerce brokers, may be very affected by the existence of replicas or near-replicas in their repositories. As such, great efforts have been applied in the development of effective and efficient methods for the removal of duplicates in large data repositories [Bell and Dravis 2006; Geer 2008].

Since this is a complex task, whose treatment requires a lot of time and processing power due to the need of comparing large amount of records, machine learning techniques have been used with success

in the resolution of the deduplication problem. However, such techniques require a training phase in which manually labeled examples are used to learn deduplication patterns. This manual process makes difficult the application of these techniques in several real situations due to the cost and time necessary to cretate the set of training examples. In this article, we propose an approach based on a deterministic technique to suggest, in an automatic fashion, training examples for a GP-based record deduplication method, making viable the use of this method in large real-world applications.

The results of an experimental study presented in Section 5 show that it is possible to use a (very) reduced amount of training examples without affecting much the effectiveness of the individuals (deduplication functions) generated in the end of the evolutionary process. For the data repositories used in this work, the utilization of 10% of positive and negative pairs of records is already enough for obtaining satisfactory results, significantly reducing the time needed for training. Moreover, a deterministic method was successfully applied for the automatic selection of training examples for the process of record deduplication using a GP-based method, helping to validate the approach proposed in this article.

Several extensions of our work can be envisioned. An experimental study using real data repositories of different domains and levels of difficulty will help to consolidate our current results. Different methods for the selection of training examples could also be tested, for example, choosing the pairs of records that lied closer to the replication identification thresholds, possibly improving the current results. Moreover, we could use different deterministic techniques for the selection of training examples allowing a comparison of different approaches for this task. Finally, a graphical interface (*GUI*) could be developed to facilitate the process of generating deduplication functions with the GP-based method. In the current version of the developed tool, the user needs to modify several files written in Python to configurate the whole record deduplication environment, which requires a good knowledge about the syntax of the language and restricts the potential use of the method to experient users. A graphical interface would make it easier for a user with less experience in Python and the record deduplication environment to use the proposed method.

REFERENCES

Baeza-Yates, R. A. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

Banzhaf, W., Francone, F. D., Keller, R. E., and Nordin, P. *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

Bell, R. and Dravis, F. Is your data dirty?: (and does that matter?). Tech. rep., Accenture Whiter Paper, 2006. Available at http://www.accenture.com.

Bhattacharya, I. and Getoor, L. Iterative record linkage for cleaning and integration. In *Proceeding of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. Paris, France, pp. 11–18, 2004.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. Adaptive name matching in information integration. *IEEE Intelligent Systems* 18 (5): 16–23, 2003.

Bilenko, M. and Mooney, R. J. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC, USA, pp. 39–48, 2003.

Carvalho, J. C. P. and da Silva, A. S. Finding similar identities among objects from multiple web sources. In *Proceedings of the Fifth ACM International Workshop on Web Information and Data Management*. ACM, New York, NY, USA, pp. 90–93, 2003.

Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. San Diego, CA, USA, pp. 313–324, 2003.

Christen, P. Probabilistic data generation for deduplication and data linkage. In *Intelligent Data Engineering and Automated Learning*, M. Gallagher, J. M. Hogan, and F. Maire (Eds.). Lecture Notes in Computer Science, vol. 3578. Springer, pp. 109–116, 2005.

Christen, P. Febrl: a freely available record linkage system with a graphical user interface. In *Proceedings of the Second Australasian Workshop on Health Data and Knowledge Management*. Wollongong, NSW, Australia, pp. 17–25, 2008.

Cohen, W. W. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems* 18 (3): 288–321, 2000.

de Carvalho, M. G., Gonçalves, M. A., Laender, A. H. F., and da Silva, A. S. Learning to deduplicate. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*. Chapel Hill, NC, USA, pp. 41–50, 2006.

de Carvalho, M. G., Laender, A. H. F., Gonçalves, M. A., and da Silva, A. S. Replica identification using genetic programming. In *Proceedings of the 2008 ACM Symposium on Applied Computing*. Fortaleza, CE, Brazil, pp. 1801–1806, 2008.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering* 19 (1): 1–16, 2007.

Fellegi, I. P. and Sunter, A. B. A theory for record linkage. *Journal of the American Statistical Association* 64 (328): 1183–1210, 1969.

Geer, D. Reducing the storage burden via data deduplication. *Computer* 41 (12): 15–17, 2008.

Gu, L. and Baxter, R. Decision models for record linkage. *Selected Papers from Australasian Data Mining Conference* vol. 3755, pp. 146–160, 2006.

Joachims, T. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

Koudas, N., Sarawagi, S., and Srivastava, D. Record linkage: similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. Chicago, IL, USA, pp. 802–803, 2006.

Koza, J. R. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, Cambridge, MA, 1992.

Newcombe, H. B., Kennedy, J. M., Axford, S., and James, A. Automatic linkage of vital records. *Science* 130 (3381): 954–959, October, 1959.

Salton, G. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

Tejada, S., Knoblock, C. A., and Minton, S. Learning object identification rules for information integration. *Inf. Syst.* 26 (8): 607–633, 2001.

Verykios, V. S., Moustakides, G. V., and Elfeky, M. G. A bayesian decision model for cost optimal record matching. *The VLDB Journal* 12 (1): 28–40, 2003.

Wheatley, M. Operation clean data. Tech. rep., CIO Asia Magazine. August, 2004. Available at http://www.cio-asia.com.

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