1999

Record Matching to Improve Data Quality

Vassilios S. Verykios

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

Report Number:
99-005

RECORD MATCHING TO IMPROVE DATA QUALITY

Vassilios S. Verykios
Ahmed K. Elmagarmid
Elias N. Houstis

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD-TR #99-005
February 1999

Record Matching to Improve Data Quality

Vassilios S. Verykios, Ahmed K. Elmagarmid and Elias N. Houstis

Department of Computer Sciences, Purdue University

West Lafayette, IN 47907-1398, USA.

Email: {verykios,ake,enh}@cs.purdue.edu

February 22, 1999

Abstract

Data Quality is defined in [TB98] as fitness for use, which implies that quality is relative to the use of data. Problems with data quality tend to fall into two categories: inconsistency among systems and inconsistency with reality. Format/syntax, semantic and value inconsistencies are representative of inconsistency among systems whereas incorrect and missing values are representative of inconsistencies with reality.

In this paper, we address the *record matching* problem which is related to value inconsistencies and incorrect or missing values. Inconsistencies related to duplicated or partially overlapping information among systems occur if changes in one system are not reflected in the other systems for various reasons such as bad design, lack of trust among systems, etc. The difficulties inherent in attempts to identify entities from different interoperating systems (as they independently evolve over time) that refer to the same real life entity are known as the record matching problem. This is a typical problem in multi-system organizations where data residing in diverse systems needs to be merged, either for assessing financial risks or for cutting down costs associated with various projects.

The methodology presented in this paper unifies a variety of techniques addressing the record matching problem, which we are considering as a classification task. The techniques used are the

following: inductive learning, clustering, fuzzy set theory, and uncertainty reasoning, all of which will improve existing methodologies with regards to the accuracy of the record matching approach as well as the computational complexity of identifying approximate duplicate records in large data sets.

1 Introduction

Many companies have multiple legacy and information systems that support their operations. These systems contain a great deal of redundant, summarized, and overlapping data objects that are often dependent on each other. Dependent objects [GKG97] may be rows in relational databases or data and functionality maintained in legacy information systems [EHKU96b]. However, the mutual consistency of data among systems, or even in the same system, is often questionable. Updates on a specific system (source system) are often not propagated to another system (target system) that contains interdependent data because of various reasons. Such a situation helps develop data quality problems that originate as a result of data dependencies. These problems are quite common in the industry today and they are the cause of significant revenue loss. In [EHKU96a], it is reported that 25% of the customers of a large telecommunication company do not receive their bills, due to bad addresses, and that 25% of field representatives' time, in some areas, is wasted because of inaccurate paired data.

Some of the approaches used to cope with these problems are listed below:

(a) *Data Edits* are computerized routines, which certify whether data values and their representations satisfy predetermined constraints. Such constraints are sometimes called business rules, and they can either be very simple or quite sophisticated. Data edits may be applied to an entire database as a screen or as a filter within a data flow.

(b) *Data Correction* matches the data against known lists (usually addresses, but sometimes master lists such as recipient databases) and ensures that all fields are tagged as good, bad, or automatically correctable. Most factory-management software systems have access to data range tables which are used to filter out-of-range data.

(c) *Record matching* determines whether two records, perhaps of different types, represent data on the same object. This process involves many value judgments and requires sophisticated software tools. Customer matching and retail house-holding are two examples where record matching can be applied. In the first application, we want to find the same customer when the customer buys a second or a third product from the same supplier, and in the second, we want to find a group of people who comprise a family, of which every member is a customer.

The methodology presented in this paper unifies a variety of techniques to deal with both the high dimensionality of the record matching problem and the uncertainty in identifying pairs of approximately duplicate records, as well as the increased complexity of solving this problem in large databases. We are considering the record matching problem as a classification task. Classification means mapping from a vector of feature measurements to a categorical class variable. The process of estimating this mapping is sometimes referred to as *supervised learning* in Machine Learning community. Typical classification problems involve large databases with many attributes per record. A wide range of problems that fit into this framework are: stock selection, targeted marketing, medical diagnosis, etc.

Intuition and human expertise (i.e, domain/background knowledge) will be very important at this step in order to capture and identify the most pertaining to the application features and to get rid of insignificant ones. This can also help in some cases during the automated induction process, to reduce the searchable space of patterns and hypotheses, and thus to increase the performance of the learning algorithm.

Even in a manually reduced feature space, generating a classification model from a large number of examples is a difficult task. On one hand, this is exemplified by the high volume of data. On the other hand, we have a large number of relationships among the predicting variables (features) and the class variable that can be uncovered by the learning algorithm for explaining the record matching process (in other words the end result of the classification task).

The high dimensionality of the problem calls for automated inductive learning classification algorithms such as decision tree inducers, instance based techniques, etc., to induce a model that captures the important characteristics of this process, by using previously seen representative examples. Feature subset selection (automatically performed by some software implementations of inductive learning techniques and for further decreasing the dimensionality of the problem given to the inducer) and pruning (to avoid overfitting after the initial model is built) are two approaches which can reduce both the time for building the model and the complexity of the induced model.

Clustering is used to decrease the number of record pair comparisons required for identifying approximate duplicate records in both the training set and the entire database. Sorting and sampling can be used to reduce the complexity of inducing the initial model that maps feature vectors of record pairs into classes. A rule transformation process can further reduce the complexity and size of the final set of rules which will be used by the inference engine to identify duplicate records in the entire database. Finally, Fuzzy set theory and certainty factors can be utilized by the reasoning mechanism to achieve higher predictive accuracy than would normally be obtained by the rule set produced by transforming the initially induced model to rules.

The organization of the paper is as follows: Section 2 discusses related work in this area and gives a brief description of our approach. Our methodology is presented in detail in Section 3. The experiments we conducted on synthetic data and the results are presented in Section 4. Section 5 concludes our discussion and refers to our future plans.

2 Background and Related Work

Estimating and/or improving the quality [WW96, WSF95, WK93] of data stored in real life databases are difficult problems because of the data volume and the variety of ways errors might be introduced in a system (data entry errors, bad design). Consequently, heuristic ad-hoc solutions are often sought to balance the computational load and the predictive accuracy.

Concerning the data quality of database answers, Motro and Rakov [MR97] have proposed a methodology in which they rely on a previously specified error model for representing possible errors in the database. Having decided upon the error model, they apply machine learning and statistical techniques to identify homogeneous areas in the database with respect to errors. After separating the database into segments of similar quality, they can measure the quality of an answer to a query or a view as a combination of the quality of the basic segments. In this way, they can compute the total quality of any area in the database that represents the answer to a selection, projection, or join query.

Finding the errors that exist in a database–a problem overlooked in the previous approach–is by itself an even harder problem than simply estimating the quality of data by assuming a known error model. An example of this problem is identifying records in a database that are duplicates of each other. This problem has been called names such as the "merge-purge" problem, the "field-matching" problem, data reconciliation, or in general approximate matching record detection.

An early method devised for addressing the problem of exact duplicate elimination uses a sorting technique which takes an entire record and uses it as the comparison key, in order for identical records to be positioned close to each other. After this step, the database is scanned sequentially and a small window of records is searched for exact duplicate records. An improvement over this technique is presented in [BW83] based on a modified merge-sort procedure. If the size of the window is small enough, then the complexity of this approach is equivalent to the complexity of the sorting algorithm.

Calling it the "sorted neighborhood method", Hernadez and Stolfo [HS98, HS95] enhance the traditional approach through the use of multiple passes (different parts of a record are used as keys in each pass for sorting) and then combining the results of all separate passes in a transitive closure step. Along the same lines, Monge and Elkan [ME97] use the entire record as a key, processing it both from left to right and from right to left, for sorting the database out. In their approach, they use a small pre-specified number of clusters (in union/find structures) that temporarily keeps sets of approximately duplicate records; they check whether the currently processed record is a member of one of the existing clusters by applying a similarity checking algorithm between the current record and the representatives of the existing clusters. If the algorithm finds that the record belongs to a cluster, it inserts the record to that cluster. Otherwise a new cluster is created, and the record is inserted in the newly created cluster. If the number of clusters exceeds the maximum amount, then an old cluster is deleted. Although the two techniques outlined above show great improvement over their predecessors with respect to both the reduction in the number of record comparisons and the efficiency of the selected structures utilized, they have two basic flaws: they lack an automatic methodology which generates the equational theory for similarity searching and they suffer from the increased complexity of applying the manually identified theory for similarity.

Cochinwala et. al. [CKLS98] demonstrate a technique to automatically generate an equational theory for record similarity detection through machine learning and statistical techniques as well as record sampling for reducing the complexity of inducing the model. Our approach demonstrates a clear improvement over the technique presented in [CKLS98]. In particular, our methodology makes use of:

1. background knowledge, feature subset selection and clustering techniques for complexity reduction,

2. sorting of records inside each cluster based on a reduced key for improving the quality of sampling along with a neighborhood-like technique to reduce the number of pairs of records which needs to be considered,

3. automatic induction of a decision tree model, pruning to avoid overfitting and transformation of the induced model to production rules for increasing the performance of the similarity detection process [Qui87], and

4. uncertainty reasoning (based on certainty factors and fuzzy set theory) for masking the errors in the predictive behavior of the model related to both record sampling and the specific values selected by the inductive algorithm for splitting up the domain of each one of the features.

3 Methodology

First, we must realize that there is no unique and accurate algorithm which identifies whether two records are matched or not because of the uncertainty of what is considered "similar" or "approximately equal" and the inability to express it in a computer readable form. For this reason, we concentrate on near-optimal solutions with respect to both complexity and accuracy. For example, if m is the complexity of an algorithm that identifies approximate duplicate records under constraints in its accuracy and N is the size of database in records, then the complexity of a method that identifies approximate duplicate records in the entire database is of complexity $O(mN^2)$.

Our methodology automatically builds a model to be used for checking record similarity with expected accuracy rate and low computational effort. At the same time, it considerably decreases the number of all-to-all record comparisons required by the brute-force approach. The core part of this methodology consists of the following phases: (i) data engineering, (ii) model induction and rule generation, and (iii) management of uncertain information. In the following subsections, we describe each one of these phases in detail.

3.1 Data Engineering

This phase includes problem specification and data pre-processing before the inductive algorithm takes over, inducing the decision tree model.

3.1.1 Feature Space and Problem Dimensionality

Various distance metrics between the same or different fields in pairs of records and various record descriptors can be used to compute a-priori probabilities for estimating the a-posteriori probability that a pair of records is or is not similar. Metrics previously used for this task include the edit distance between strings, the length of the field descriptors, the Euclidean distance between numerical features, and trigrams. The entire set of all these metrics constitutes the feature space of the record matching classification task.

3.1.2 Data Conditioning

A pre-processing phase is necessary to prepare the data and condition them, as well as provide us with an approximate estimation of the quality of the data. This is almost always a domain dependent process and can facilitate many of the later phases. Some of the issues that must be addressed at this point are outliers, null values, missing values, as well as the application of some domain rules.

3.1.3 Clustering

A clustering technique can bring approximate records close to each other. This step is very helpful when it is known that some of the fields in the considered databases are error-free. For example, if there is a-priori knowledge that the field "sex" with possible field values "male" and "female" and the field "marital status" with possible field values "yes" and "no" are clean, then we can very easily cluster the set of data records into four clusters, e.g., the first cluster contains all the records for unmarried males, the second cluster contains the married males, etc. Since we know that the values corresponding to these fields are correct, there is no possibility of records belonging to different clusters being approximate duplicates. We assume that clustering will not create small sets, since the sets still contain a very large number of records, but that it will facilitate the later phases since no comparisons will be required for detecting known to be clean fields.

3.1.4 Sampling, Sorting, and Labeling

The data needs to be sampled in order for the learning process to be efficient and flexible. Based on domain knowledge, we can use certain parts of the sampled records and generate keys that can be used for sorting these records. After the records have been sorted we can use a neighborhood-like approach to reduce the number of record comparisons required for similarity checking.

3.2 Model Induction and Rule Generation

In this phase, a model is built and then transformed into a rule set. The rule set will comprise the rule base of an expert system that will be used for identifying similar records in the entire database.

3.2.1 Model Induction

Having collected the information from the previous step, we can use a data mining technique that induces models from examples. One of these techniques is a decision tree classifier [Qui96] that induces decision trees from examples for prediction purposes. A decision tree inducer builds trees that can be used for predicting the label of a pair of previously unseen records, based on the values of the metrics (features or combination of features) computed for this pair of records. The non-leaf nodes of a decision tree are labeled by the name of one of the metrics computed in the training phase, and the edges originating from these nodes are labeled with values or ranges of values corresponding to the metric that labels the node. The leaf nodes are labeled by the category that the majority of the pairs of records covered by each node belong to. In our case, a non-leaf node may be labeled by the edit distance between the strings corresponding to the first name field of each record; the outgoing edges from this node may be labeled by possible values or groups of values that can be attained by this metric; and a leaf node can be labeled with the value "yes", designating that the class of the pair of records falling into this leaf node belongs to the "matched" records. Pruning is one of the traditional techniques used to avoid overfitting because usually the initially induced model fits noise i.e., correctly classifies examples that have been corrupted by noise. Pruning produces trees of smaller size than the initial tree with higher predictive accuracy.

3.2.2 Generation of Production Rules

The decision tree induced in the previous phase has to be transformed into a set of production rules that will constitute the rule base of the expert system. A decision tree can be easily transformed into a rule set by converting each path of the tree into a rule as follows: (a) the internal nodes and their output branches are converted into conditions of the antecedent ("if-part") of the rule; and (b) the leaf nodes are converted into the consequent ("then-part") of the rule.

As a side effect, we try to increase the accuracy of the induced model and, at the same time, decrease its complexity and size for efficiency reasons. For further improvement of the quality of the rule set, we adopt the method proposed in [Qui87] where a 2×2 contingency table is built for each of the rules after removing one condition from its antecedent. Confidence or certainty factors offer a simple tool for representing uncertainty. In expert systems, they have been used to denote confidence that stated facts and rules do indeed describe real-world objects or relationships. By computing a certainty factor, we can tell whether the eliminated from the antecedent tested condition can be completely dropped without affecting the accuracy of the rule. The number of rules generated this way is always smaller than the number of rules generated by initially transforming the patterns in the decision tree to rules. One of the consequences of dropping conditions from the antecedents of the rules is that an input case can be matched against more than one rules. In this case, the expert system puts all the matched rules into its agenda and fires the one with the highest certainty factor.

Furthermore, the generated rule set can be optimized in the following sense. For each rule r in the rule set, we count the misclassifications produced by the rule set $R - \{r\}$ in the training set. If the reduced rule set $R - \{r\}$ gives at least as much accuracy as the entire set R, then the rule r is removed from the rule set R.

3.3 Dealing with Uncertainty

Improving the quality of the rule base results in both reducing the processing time required by the expert system for the duplicate detection process and increasing the accuracy of the entire process.

3.3.1 Fuzzification

Having used a sample set of cases as a training set for the inductive algorithm, we expect that, even though the generated rules capture most of the important relationships between pairs of records' features and classes labels, the ranges of values produced by the particular splittings selected by the classification algorithm should be relaxed in some way. Our solution to this problem is to fuzzify the domain of each range of values for each variable. Classically, sets are crisp in the sense that an element either belongs to a set or is excluded from it. Fuzzy set theory generalizes the classical characteristic function of a set, which maps the elements of the universe to the set $\{0, 1\}$, to a function that maps the universe to the interval $[0, 1]$. For example, if a rule that reads as follows is generated:

```
if A<10 then class=C1 with certainty factor=0.9
else if A>=10 class=C2 with certainty factor=0.85
```

for each one of the two ranges for the variable A, we can impose a membership function which assigns to each member of the ranges of values a degree of membership to the fuzzy set. For example, `A=1` may be assigned a degree of membership equal to 1.0, while `A=9` may be assigned a degree of membership equal to 0.9 to the fuzzy set `small A` which corresponds to the crisp set $A < 10$, for specifying our uncertainty regarding the splitting of the variable's A domain by the value 10. The degree of membership of a specific value to the fuzzy set will be used by the expert system for appropriately modifying the certainty factor of each of the matched rules on the fly. For example, if we consider the same rule as above, and the value of the variable A for a specific case is 9, the antecedent of the rule is satisfied and will be placed into the agenda, but the certainty factor of the rule will be 0.95×0.9, where 0.95 is the degree of membership of the value 9 to the fuzzy set `small A`, and 0.9 the certainty factor of the rule. So, we can see that by fuzzifying the ranges of values generated by the splitting of the domains of variables into subranges, we can modify on the fly the certainty factors of the rules and thus make the rule base less sensitive to the particular splitting values.

3.3.2 Complementary Rule Set

If the certainty of a rule is not satisfactory, then further tests should be applied in order to determine whether two records are or are not similar. A good indication for low certainty is when the certainty of the majority class label of a rule falls under the certainty factor for its corresponding minority class label, or the certainty of the majority class falls under a user specified limit. An automatic solution for this problem would be to project the sample set to a subset of variables, excluding the ones that originally used by the induction algorithm, and let the inducer build a new tree for these attributes. The knowledge produced this way can be used as an additional knowledge base that the expert system can use to determine similarity of records. Manual intervention at this stage might be necessary since we expect that will be difficult or impossible for the system itself to identify very complicated or even (previously) unseen cases in the training set. We can be optimistic though that these cases are rare, and even if this is not the case, this is the only safe way we can handle such cases.

After the knowledge bases have been produced, we have the expert system run for the entire set of records in the cluster. The entire process described above (not including the clustering) must be repeated for all the clusters.

4 Method Implementation and Experiments

In order to test our methodology, we generated synthetic data. The results we present are based on a set of records taken from the "Student and Staff Telephone Directory" of Purdue University. Each record contains the first, middle, and last name, the home address, office address, and telephone number of an individual. Each record has a system specific key which automatically assigned by the system. By replicating the records and generating errors in a random way, we can simulate a variety of real life scenarios. Each replicated record maintains its original key, so that during the matching training process, labels are automatically assigned to the pair of records regarding this similarity. For numerical fields, we evaluate the numerical Euclidean distance, and for character string fields, we evaluate the well-known "edit-distance" metric, by implementing the minimum edit distance algorithm [Man89] which is of complexity $O(mn)$ where m and n are the sizes of the compared strings.

The pre-processing phase consists of removing specific parts from the fields (such as Rd., Street, etc.) and sorting the strings inside each field. Because the data we generated is synthetic, we did not perform any clustering since writing the code for implementing this phase is quite straightforward and will not add any significance to the experiment. Alternative techniques can be used for clustering based on numerical fields by using public domain software like AutoClass [CS96] developed by NASA Ames Research Center.

The feature vectors we use for the learning process consist of the edit distances between the names and home addresses, the length of the strings for the names and addresses, and the differences between the lengths of the names and addresses.

As we pointed out earlier, the data mining technique we are using for building the classification model is a decision tree classifier. We selected the ID3 [Qui86] algorithm, which is a top-down inducer of decision trees. The implementation of this algorithm is from the MLC++ library [KSD96].

Figure 1 demonstrates the decision tree corresponding to the numbers in the first row of Table 1

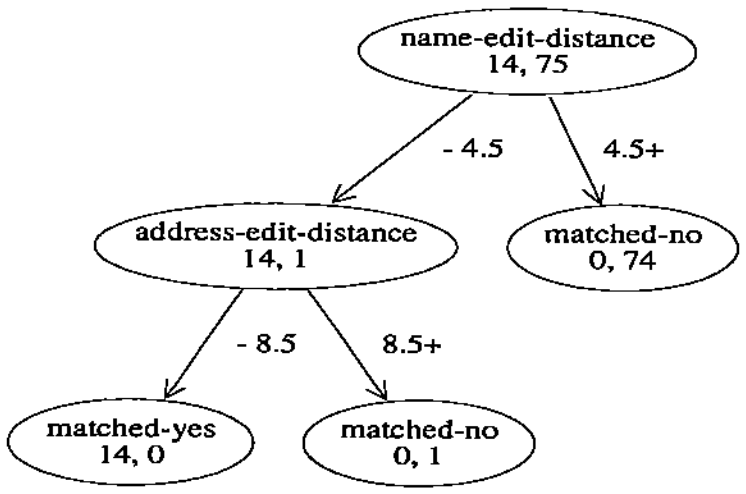

Figure 1: Decision tree generated using 89 cases as a training set.

Training Set	Test Set	Correctly classified	Incorrectly classified	Percentage Error
89	152	146	6	3.947
177	303	288	15	4.950
264	453	433	20	4.415
350	602	580	22	3.654
435	750	726	24	3.200
519	897	874	23	2.564
602	1043	1016	27	2.594

Table 1: Sizes of training and test sets and accuracy achieved by the induced decision tree.

while Figure 2 shows a more complicated decision tree corresponding to the last entry in the same table. Figure 3 shows a geometric representation of the classes "1" for "matched-yes" and "2" for "matched-no" in the 2D space of the name-edit-distance (x-axis) and address-edit-distance (y-axis). Even though the error decreases as the number of training set becomes larger and larger, the complexity of the induced models is higher which gives rise to increased computation times by the expert system.

The feature vector given to the ID3 is listed below:

```
name-edit-distance:        continuous

address-edit-distance:     continuous

length-name1:              continuous

length-name2:              continuous
```

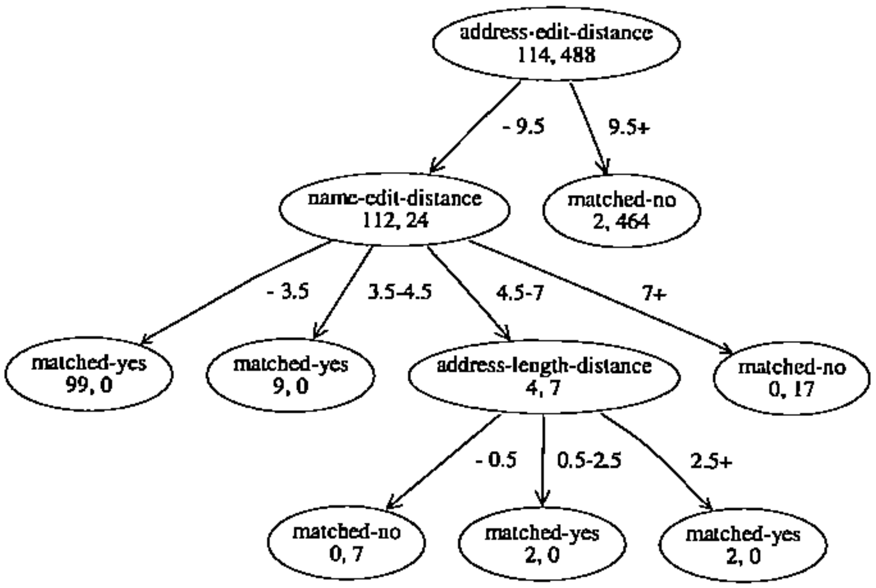

Figure 2: Decision tree generated using 602 cases as a training set.

```
name-length-distance:         continuous

address-length1:              continuous

address-length2:              continuous

address-length-distance:      continuous

matched-yes, matched-no       classes
```

and the 6 misclassified examples from the training set are listed below:

```
Incorrect classifications ...

5, 0, 11, 10, 1, 20, 20, 0, matched-yes.

5, 1, 14, 10, 4, 19, 20, 1, matched-yes.

6, 1, 10, 14, 4, 16, 15, 1, matched-yes.

5, 4, 14, 14, 0, 15, 19, 4, matched-yes.

5, 0, 14, 17, 3, 20, 20, 0, matched-yes.

6, 2, 13, 14, 1, 20, 20, 0, matched-yes.
```

By transforming the decision tree patterns from Figure 1 into production rules, we get the following

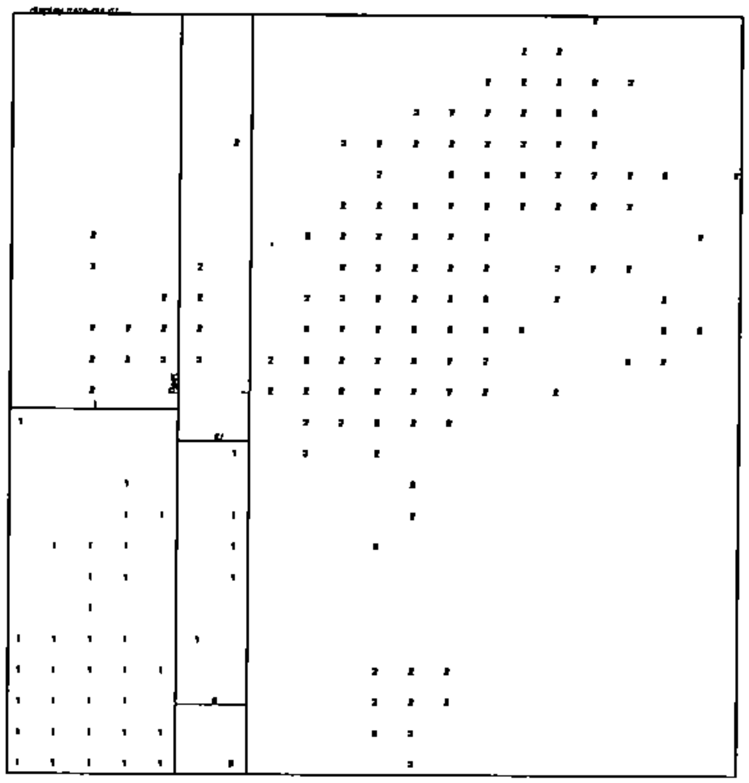

Figure 3: Pictorial representation of the distribution of classes in the 2D space. The values of the `name-edit-distance` are depicted in the x axis and the values of the `address-edit-distance` are depicted in y axis. Cases represented as "1" belong to the "matched-yes" class, while cases represented as "2" belong to the "matched-no" class.

rule set:

```
if  name-edit-distance < 4.5 and address-edit-distance < 8.5

then class=matched-yes with certainty factor 1.0

if  name-edit-distance < 4.5 and address-edit-distance >= 8.5

then class=matched-no with certainty factor 1.0

if  name-edit-distance >= 4.5

then class=matched-no with certainty factor 1.0
```

By applying the two-step approach for improving the quality of the rule set, we are left with the following reduced rule set which has exactly the same predictive accuracy as the one above:

```
if  name-edit-distance < 4.5
```

```
then class=matched-yes with certainty factor 0.933
if  name-edit-distance >= 4.5
then class=matched-no with certainty factor 1.0
```

By exploiting fuzzification in the ranges of values generated by the splittings of the decision tree inducer, we can further improve the accuracy. For example, the first feature in the cases misclassified (see incorrect classifications above) is the value for the name-edit distance. If we check all the values for this feature, we will see that the misclassified cases have values that are very close to the splitting values 4.5.

By setting a certainty factor's cut-off value equal to 0.6 and by fuzzifying the domain `high name-edit-distance` and giving degree of membership to the numbers 5, 6 equal to 0.5, 0.7 respectively, then the certainty of the matching rule (which in this case is the second rule in the reduced rule set) is below the cut-off value of 0.6. For all the cases having the value of 5 as their name-edit-distance, further tests must be applied in order to decide if they are similar or not. (Of course, some records already correctly classified will be exposed to a second similarity searching procedure, but this should be a very small part of the records that fall under each leaf).

By appropriately deploying the features that have not been utilized by the decision tree inducer in the first place or even a domain expert, we can achieve even higher accuracy than the rate achieved initially, even with a simpler (less conditions) and smaller (less number of rules) rule set.

The expert system shell we are using as the inference engine for applying the rule set to the entire database (and in particular to the test set) is CLIPS [Gia91] from NASA Johnson Space Center. The entire system has been coded in Perl [Ous98].

By applying our methodology to all the sets listed in Table 1, as well as to larger data sets, we have achieved 20% – 40% increased accuracy and reduced computation times by 30%.

5 Conclusions

Data mining techniques (clustering and classification) have been applied to the problem of approximately duplicate record detection. Certainty factors and fuzzy set theory have also been utilized for increased accuracy in domains where uncertainty is a major problem. Expert system technology is a promising vehicle to provide solutions, even under incomplete, imprecise, or uncertain information. Sampling can also be very handy in domains where an overabundance of information is a fact.

Our solution shows that great improvements can be made towards the automation of the knowledge acquisition phase for the similarity searching procedure, by combining all these techniques in a single framework, and at the same time limits the human intervention to a minimum, but does not exclude it.

The selection of appropriate model parameters and functions that convey the peculiarities in the data being processed is of big importance in the entire process for achieving increased accuracy. The tuning of these "application parameters" must be done and validated before the process of building/inducing models. We plan on applying our methodology to real life databases and comparing or improving it by considering alternative approaches for implementing the various steps such as using learning algorithms that induce rule sets instead of decision trees and comparing the results.

References

[BW83] D. Bitton and D. J. Witt. Duplicate record elimination in large data files. *ACM Transactions on Database Systems*, 8(2):255–265, 1983.

[CKLS98] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha. Efficient Data Reconciliation. Bellcore Research, February 1998.

[CS96] Peter Cheeseman and John Stutz. Bayesian Classification (AutoClass): Theory and Results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press/MIT Press, 1996.

[EHKU96a] A.K. Elmagarmid, B. Horowitz, G. Karabatis, and A. Umar. Issues in Multisystem Integration for Achieving Data Reconciliation and Aspects of Solutions. Bellcore Research, September 1996.

[EHKU96b] A.K. Elmagarmid, B. Horowitz, G. Karabatis, and A. Umar. Metrics for Data Reconciliation: An Initial Analysis. Bellcore Research, September 1996.

[Gia91] J. C. Giarratano. *CLIPS User's Guide, Version 5.1.* NASA Lyndon B. Johnson Space Center, 1991.

[GKG97] D. Georgakopoulos, G. Karabatis, and S. Gantimahapatruni. Specification and Management of Interdependent Data in Operational Systems and Data Warehouses. *Distributed and Parallel Databases*, 5(2):121–166, 1997.

[HS95] M. A. Hernadez and S. J. Stolfo. The merge-purge problem for large databases. In *Proc. of the 1995 ACM SIGMOD Conference*, pages 127–138, 1995.

[HS98] M. A. Hernadez and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Journal of Data Mining and Knowledge Discovery*, 1(2), 1998.

[KSD96] Ron Kohavi, Dan Sommerfield, and James Dougherty. Data mining using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence.* IEEE Computer Society Press, 1996. `http://www.sgi.com/Technology/mlc`.

[Man89] Udi Manber. *Introduction to Algorithms.* Addison-Wesley Publishing Company, 1989.

[ME97] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997. to appear.

[MR97] A. Motro and I. Rakov. Not all answers are equally good: Estimating the Quality of Database Answers. In T. Andreasen et al., editor, *Flexible Query-Answering Systems*, pages 1–21. Kluwer Academic Publishers, 1997.

[Ous98] John K. Ousterhout. Scripting: Higher-Level Programming for the 21st Century. *Computer*, 31(3):23–30, 1998.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Qui87] J. R. Quinlan. Generating Production Rules From Decision Trees. In *Proc. 10 Tenth International Joint Conference on Artificial Intelligence*, pages 304–307, 1987.

[Qui96] J. R. Quinlan. Learning decision tree classifiers. *ACM Computing Surveys*, 28(1):71–72, 1996.

[TB98] Giri Kumar Tayi and Donald P. Ballou. Examining data quality. *Communications of the ACM*, 41(2):54–57, 1998.

[WK93] R.Y. Wang and H.B. Kon. Towards Total Data Quality Management (TDQM). In R.Y. Wang, editor, *Information Technology in Action: Trends and Perspectives.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[WSF95] R.Y. Wang, V.C. Storey, and C.P. Firth. A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):623–640, 1995.

[WW96] Y. Wand and R.Y. Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, 1996.