# Automating the approximate record-matching process

**ARTICLE** *in* INFORMATION SCIENCES · AUGUST 2002

Impact Factor: 3.89 · DOI: 10.1016/S0020-0255(00)00013-X · Source: DBLP

Vassilios S. Verykios
Hellenic Open University
**106** PUBLICATIONS **2,923** CITATIONS

SEE PROFILE

Ahmed K. Elmagarmid
Qatar Computing Research Institute
**286** PUBLICATIONS **8,029** CITATIONS

SEE PROFILE

# Automating the Approximate Record Matching Process

Vassilios S. Verykios and Ahmed K. Elmagarmid

Computer Sciences Department

Purdue University

West Lafayette, IN 47907-1398

June 15, 1999

### Abstract

Data Quality has many dimensions one of which is accuracy. Accuracy is usually compromised by errors accidentally or intensionally introduced in a database system. These errors result in inconsistent, incomplete, or erroneous data elements. For example, a small variation in the representation of a data object, produces a unique instantiation of the object being represented. In order to improve the accuracy of the data stored in a database system, we need to compare them either with real-world counterparts or with other data stored in the same or a different system. In this paper we address the problem of matching records which refer to the same entity by computing their similarity. Exact record matching has limited applicability in this context since even simple errors like character transpositions cannot be captured in the record linking process. Our methodology deploys advanced data mining techniques for dealing with the high computational and inferential complexity of approximate record matching.

## 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 interdependent [GKG97]. Lack of a common data model, errors in data flows, errors during data entry, or situations where updates are not reflected into the database cause inconsistencies to arise

[WSF95, WW96, MR97]. These inconsistencies are common in systems today and are the cause of significant revenue loss. In [EHKU96], it is reported that up to 25% of customers records are erroneous in a typical billing system.

Existing techniques 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 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 master lists and tags all fields either as good, bad, or automatically correctable. For example, 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 the same object. This process involves 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 improves upon existing techniques used for record matching. A brute-force approach for matching records is of quadratic order with respect to the number of records in the database. This can become a bottleneck for very large databases. The high dimensionality of the problem calls for heuristics to cut down the search space and shortcuts similar to the ones almost unconsciously employed by a human filing clerk.

The main contribution of our methodology is the complete automation of the knowledge acquisition for the record matching process. The comparison space, resulting from economically comparing records in a sample drawn from the original database, is clustered, and the clusters are described to produce predictive patterns. The so generated patterns are simplified and used as a rule base for matching records in the entire database. The results of our initial experiments indicate that the proposed methodology improves the accuracy of the matching process and reduces its high complexity.

The organization of the paper is as follows: Section 2 provides background information with respect to approximate record matching. Section 3 discusses related approaches used for record matching and gives a brief description of how our approach fits into the big picture. A record matching methodology is presented in Section 4 and an example is given in Section 5. Concluding remarks are made in Section 6.

## 2 Background

The idea of linking records is very simple. Record linking or matching means bringing together information about the same entity. The two principal steps in the linking process are to search for potentially linkable pairs of records (searching step) and to decide whether or not a given pair is correctly matched (matching step). For the *searching step*, the aim must be to reduce the number of failures to bring potentially linkable records together for comparison. The technique to be applied must address this problem without resorting to excessive amounts of additional searching. For the *matching step*, the problem is that of enabling the computer to infer and apply the rules of judgment by which a human clerk would decide whether or not a pair of records relates to the same entity when some of the identifying information agrees and some disagrees.

In the product space of two tables, a *match* is a pair that represents the same entity, and a *non-match* is a pair that represents two different entities. With a single database, a *duplicate* is a record that represents the same entity as another record in the same database. Rather than regard all pairs in the product space of two databases, or the space of a single database, it may be necessary to consider only those pairs that agree on certain identifiers or *blocking criteria*. Blocking criteria are sometimes called sort keys. *Missed matches* are those false non-matches that do not agree on the set of blocking criteria.

*Matching variables* such as common record identifiers like names, addresses and code numbers like Social Security Number, are used to identify matches. The vector that keeps the values of all the attribute comparisons for a pair of records is called *comparison vector*. The set of all possible vectors is called *comparison space*. A record linkage decision rule is a rule that designates a pair either as a *link*, a *possible link*, or a *non-link* based on the information contained in the comparison vector. *Possible links* are those pairs for which identifying information is not sufficient to determine whether a pair is a match or a non-match. Typically, clerks review possible links and decide their match status. *False matches* are those non-matches that are erroneously designated as links by a decision rule. *False non-matches* are either (i) matches designated as non-links by the decision rule as it is applied to a set of pairs or (ii) matches that are not in the set of pairs to which the decision rule is applied. Generally, link/non-link refers to the designations under decision rules and match/non-match refers to the true status.

# 3 Related Work

Identifying similar records in various databases is a computationally intensive and difficult task. For large databases storing millions of records, a pair-wise comparison of all the records is not feasible. This implies that a technique for record linking has to be both computationally efficient and intelligent enough. In the following paragraphs, we describe the related work from the viewpoint of the searching and matching steps of the record linking process.

## 3.1 Searching Process

In the case of the searching step, errors in the form of failures to bring potentially linkable pairs of records together for comparison could be reduced to zero simply by comparing each record with all the other records. Where the files are large, however, such a procedure would generally be regarded as excessively costly in terms of the enormous numbers of wasted comparisons of pairs of records that are not matched. For this reason, it is usual to order the records in the database by using identifying information that is common to all the records [New67]. The ordering can be done based on the information that is kept in the key, or in some other combination of record fields. In the exact matching, sorting the file can be used to reduce the complexity of identifying duplicate records [BD83]. In the approximate record matching, various compression codes (i.e., phonetic) can be used to efficiently overcome the failures in bringing potentially linkable pairs of records together for comparison. There are a number of systems for doing this, the most common of which is known as the Russell Soundex code. This is essentially a phonetic coding, based on the assignment of code digits which are the same for any of a phonetically similar group of consonants.

Another technique for cutting down the number of unwanted comparisons in the approximate record matching is to scan the database by using a fixed size window and check for matches by comparing every pair of records that falls inside that window [HS98] assuming that the records are already sorted. This process has been augmented by the same authors with multiple passes through the database (sorted neighborhood approach) by using different sort keys in each pass and a transitive closure phase for combining the results of independent passes. The transitive closure step has been independently identified by the authors in [ME97]. Another approach is to scan the database by comparing only those records that agree on a used-defined key which for example can be the key used for sorting the records. This technique is known as *blocking.* Another approach proposed in [ME97] is the use of union/find structures for keeping track of clusters of approximately duplicate records incrementally.

With respect to the searching phase, the proposed methodology makes use of a combination of a multi-pass sorted neighborhood approach based on some pre-selected keys along with a standardization of the values kept in a subset of fields. This standardization is similar enough to the coding schemes mentioned above. In particular, before sorting starts, a pre-processing phase is applied to the records by using public domain software for standardizing lists containing name and address information.

### 3.2 Matching Process

When pairs of records are brought together for comparison, decisions must be made as to whether these are to be regarded as linked, not linked, or possibly linked, depending upon the various agreements and disagreements of items of identifying information. For example, if we are linking person records, a possible measurement would be to compare family names on the two records, and assign the value of 1 for those pairs where there is agreement and 0 for those pairs where there is disagreement. These measurements will yield a vector of observations on each record pair.

Most of the work proposed by statisticians have been influenced by the pioneering work of Fellegi and Sunter [FS69]. Their paper documents the derivation of a test statistic and a critical region for deciding whether or not a pair of records is a match. In addition, it discusses some of the assumptions necessary for practical applications and describes approaches for estimating the field matching probabilities which are used to calculate the test statistic.

Authors in [HS98] suggest the use of an equational theory, manually acquired by domain experts, to be codified as a rule base of an expert system. Authors in [ME97] suggest the use of a domain-independent algorithm for detecting approximately duplicate records. Authors in [CKLS98] suggest the training of a prediction model by generating a table of prospective matches to be used as a training set, and the reduction in the complexity of the model by statistical techniques.

Our approach uses a clustering technique to automatically identify *comparison classes* in the comparison space of a sample set of records. The space generated by the comparison vectors along with the corresponding classes identified during clustering is automatically reduced by a *feature subset selection* process that minimizes the number of vector components used for building the predictive model which determines the linking status of any pair of records in the database. The induced model is automatically transformed into a set of production rules and is simplified for becoming more accurate.

## 4 Methodology

The proposed methodology tries to overcome the computational inefficiencies of all the previous approaches by automatically building a model that captures the most important patterns that exist in the data. Well established machine learning techniques can be adequately utilized in this framework for building a minimal model that fits the data very well.

There are four phases in our methodology that are listed below and are explained in the subsequent subsections.

- Data pre-processing of the records for dealing with various semantic inconsistencies with the data as well as problems of individual records like missing values, null values, outliers, etc. Data standardization is also a very important task for the preparation of the data for the later phases.

- Sorting of individual records based on values of a pre-specified key (blocking variable), and sampling of the records. Scanning of the records in the sample set and pair-wise comparison of all records falling inside a constant size window. The result of this phase is the generation of the comparison vectors for the record pairs in the sample set.

- Use of a clustering algorithm to identify clusters in the comparison space of the sample set, and use of the clusters identified as labels of the comparison vectors for building a model by using a decision tree inducer.
- Transformation of the decision tree into a rule set and application of a simplification technique for reducing the size of the rule set and/or improving its predictive accuracy. The final rule set will be used as the linkage rule set for deciding upon the linking status of all the pairs of records falling inside a constant size window in the entire sorted data set.

## 4.1 Data Pre-processing

Preparing the data for the later phases is an important and domain specific task. For example, the conversion of the characters in the data stream into either lowercase or uppercase helps to avoid any variations that may exist due to case differences between various data elements. In many systems, records are stored as discrete fielded entries. Although this provides a powerful technique for modeling during the analysis, the individual components must be combined in such a way as to represent each entity uniquely. For instance, fields involving last name and first name or street number, city, state, and zip code need to be concatenated. Field values usually contain many different types of extraneous characters. Whenever possible, they should be removed from the data stream.

Moreover, it is sometimes useful to reduce the information represented in a field to a minimum number of categorical values. This allows for conveying certain types of information without the overhead of representing every single unique value. A standard unit must be adopted for each field appearing in the data set. For example, if there is a field containing measures of distance, a standard conversion to one unit must be adopted for all values represented in the database. Because the actual processing of the data by the record linking technique, as we discussed earlier, requires valuable computational time, it is beneficial to restrict the data set to a minimal one that exhibits certain quality characteristics. Records containing empty or null fields that are critical to the analysis can usually be removed. Also value boundary violations, that result in bad or dirty data can be removed.

We should point out, that the data pre-processing phase is heavily dependent on the domain and the condition of the in-flow data. If for example null values are common for the entire data set, then it might be better to project on all the other attributes but the one with the null values instead of discarding the entire data set. In addition, some boundary violation problems can be easily resolved, if we can find out the semantics of the values represented in the records. For example, a value of 99 for the month field may represent total values for all months in the specified year.

Another very important pre-processing task before any attempt is made to link records, is the standardization of the data. Appropriate parsing of name and address components is the most crucial part of computerized record linking. Without it, many true matches would erroneously be designated as non-links because common identifying information could not be compared. The basic ideas of standardization are (i) to replace the many spelling variations of commonly occurring words with standard spellings such as a fixed set of abbreviations or spellings and (ii) to use certain key words that are found during standardization as hints for parsing subroutines. In standardizing names, words of little distinguishing power as "Corporation" or "Limited" are replaced with consistent abbreviations such as "CORP" and "LTD", respectively. First name spelling variations such as "Rob" and "Bobbie" might be replaced with a consistent assumed original spelling such as "Robert" or an identifying root word such as "Robt". Standardization of addresses operates like standardization of names. Words such as "Road" or "Rural Route" are typically replaced by appropriate abbreviations.

### 4.2 Sorting, Sampling and the Sorted Neighborhood Approach

Before we are able to efficiently process the records we need to sort them. When we sort the data, we actually generate a linear ordering of them. In order for this technique to be effective, domain knowledge is very important. Domain experts should sort the data based on a-priori knowledge that certain fields or parts of fields have a good discriminating power for the record linking process. We expect that these parts should in some general sense be clean, or to a large extent are not affected by the various kinds of errors introduced in the system. If this is not the case, multiple passes might be required for achieving good results. The reason that multiple passes for sorting purposes through the data help in resolving this problem is the following. By using a blocking or a sorted neighborhood technique, records having different values for the blocking variable or falling outside a certain window will never be considered or tested for linking. If, for example, there is an error in the sort key of a record, then this record will be placed incorrectly far away from its correct place in the sorted file hindering in this way the identification of true matching records. By selecting different sort keys in different passes, there is a high probability that in one of these passes the record will be located in the "right" place and the identification of the linked records will be achieved. Scanning the data many times comes with a heavy duty of decreased performance, and it must always be considered as a function of the available computational resources, the available time and cost constraints that must be met in a business environment, and the balance between two kinds of errors in the statistical sense. The first one identifies a pair of records as a link even if the records do not actually match, and the second one identifies a pair of records as a non-link even if the records do match. Depending on the impact of these two errors, we must decide for the selection of sort keys and the number of passes.

After the records have been sorted, we can easily extract a sequential subset of records from the database as a sample and use it for further processing. The reason that we do not actually need to apply any involved algorithm for sampling is that the most important events in the data are restricted to small neighborhoods of contiguous records. Selecting the appropriate size of the sample set is another issue, but we expect that it can be easily resolved experimentally. The sorted neighborhood technique suggested by Hernadez and Stolfo [HS98], will be applied now to the sample set of records. The records falling within a range of size $w$ from the location of a reference record in the sorted sample file will be compared among each other and this process will be repeated for all the records in the sample database. Notice that the value of $w$ for this phase must be bigger than the corresponding value of $w$ when the entire database is searched for linkable records. This is because the training process in the next phase requires both positive (linked) and negative (non-linked) examples for correctly learning the underlying domain model. The comparison vector generated by a record pair comparison will be holding information related to the closeness of the various attributes in the pair of records. The comparison of any such pair of records can be viewed as a set of outcomes, each of which is the result of comparing a specific attribute from one record with the same attribute in the other record. Outcomes may be defined as specifically as desired. For example, one might define an outcome of a comparison to be simply that the attributes agree or disagree. Or, one might define the agreement outcome more specifically, based on the possible values that attribute can take. “Comparison of attributes” is usually interpreted to mean that the same attribute is recorded on each record and that they can be compared directly. However, it is possible to “compare” different attributes which are known to be correlated or to use information related to the record descriptors.

## 4.3 Clustering and Classification

The comparison space, as we said earlier on, consists of comparison vectors containing information which is related to the differences between fields of a pair of records. The particular components of these vectors contain values which can be either of continuous or discrete type. A value of a continuous attribute in the comparison vector is always a real number, whereas the value of a discrete attribute is one of a small set of possible values. The next step is how to determine the linking status of a pair of records based on the information kept in its comparison vector. Although we can be sure that any vector must correspond to one of the three possible classes, the *link*, *non-link* and *possible link* one, there is no easy way to assign these classes or labels to the comparison vectors. If we are able to assign these classes to the comparison vectors, we can use the comparison vectors along with their corresponding classes for building a model for predicting the class of any comparison vector, and this model can be used as a record linkage rule set.

In order to solve the problem of labeling the comparison vectors, we consider a slightly different problem. Instead of trying to immediately assign labels to comparison vectors, we use a technique to identify clusters of comparison vectors, and then map these clusters to the three labels corresponding to the record linking status. *Clustering*, is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data. The categories may be mutually exclusive and exhaustive, or consist of a richer representation such as hierarchical or overlapping categories. A clustering technique can be easily applied to the comparison space of the record linking methodology in order to identify clusters of vectors.

Combining the cluster values identified by the clustering algorithm with their corresponding comparison vectors, we can create a set of training vectors to be given as input to a data mining technique that induces models from labeled vectors. The technique used for inducing the model is a decision tree inducer [Qui86, Qui96] that generates 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 vector components computed for this pair of records. The set of comparison vectors with known classes from which the decision tree will be induced is called the *training set.* Other collections of comparison vectors not seen while the tree was being developed are known as *test sets* and are commonly used to evaluate the performance of the tree. The non-leaf nodes of a decision tree are labeled by the name of one of the vector components computed in the training phase, and the edges originating from these nodes are labeled with values or ranges of values corresponding to the component that labels the node. The leaf nodes are labeled by the linking status that the majority of comparison vectors 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 of each record; the outgoing edges from this node is labeled by possible values or groups of values that can be attained by this component; and a leaf node can be labeled by the value "matched-yes", designating that the majority class of the comparison vectors covered by the particular leaf node belongs to the "matched" status.

*Pruning* is one of the traditional techniques used to avoid over fitting because usually the initially induced model correctly classifies examples that have been corrupted by noise. Pruning produces trees of smaller size than the initial tree with higher predictive accuracy. *Feature subset selection* is another important technique that affects the complexity of the record linking process in two ways. It is well understood that not all vector components have the same discriminating power with respect to the class. By including in the induced model only this subset of components that have the highest correlation with the class label, the induction algorithm applies an implicit form of feature selection. As the computations required for determining the values of each component of the comparison vector is an intensive one, we expect high payoffs from the feature selection mechanism with respect to cutting down the overall processing time. New techniques for feature selection can be used before an induction algorithm is applied, for reducing the dimensionality of the feature space, especially for these kinds of algorithms that use the entire feature (comparison) vector for inferencing purposes like instance based techniques. With this respect, our methodology is not restricted by a particular kind of induction

|  | Class $c$ | Not<br>Class $c$ |
|---|---|---|
| satisfies $X_i$ | $sc$ | $s\overline{c}$ |
| does not satisfy $X_i$ | $\overline{s}c$ | $\overline{s}\overline{c}$ |

Table 1: A $2 \times 2$ contingency table used for the simplification of the decision trees where $sc$ is the number of these cases that satisfy $X_i$ and belong to the class $c$, $s\overline{c}$ is the number that satisfy $X_i$ but belong to some class other than $c$, and so on.

technique, and many alternatives can be explored and tested for their predictive accuracy.

### 4.4 Decision Tree Transformation and Simplification

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. Merely rewriting a tree as a collection of these equivalent production rules would not represent any simplification at all. For further improvement of the quality of the rule set, we adopt the method proposed in [Qui87b, Qui87a]. The process has two stages: individual production rules are first generated and polished, and then the rules produced are evaluated as a collection. The first stage examines each production rule to see whether it should be generalized by dropping conditions from its left-hand side. Let $X_i$ be one of the conditions and consider those comparison vectors in the training set that satisfy all the other conditions in the rule. With respect only to those cases, the relevance of $X_i$ to determining whether a comparison vector belongs to a certain class $c$ (given that the other conditions are satisfied) can be summarized by the $2 \times 2$ contingency table which is shown in Table 1.

A contingency table is a form of presentation of grouped data. In the simplest case, a group of items may be classified into just two groups, according to, say, presence or absence of a certain characteristic. The categories may be qualitative or quantitative. When there is only one characteristic according to which data are classified, we get a one-way table. If there are two ways of classification, the table is called a two-way table.

The final step in this first stage is to estimate a *certainty factor* for the simplified rule. If the left-hand side of a rule is satisfied by $V$ cases in the training set, $W$ of which belong to the class indicated by the right-hand side, the certainty factor of the production rule is taken as $(W - 1/2)/V$. By computing a certainty factor, we can estimate 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.

The second stage of the simplification process looks how well the rules will function as a set. This evaluation depends on the way the rules will be used. A simple strategy can be adopted: To classify a case, find a rule that applies to it; if there is more than one, choose the rule with the higher certainty factor; if no rule applies, take the class by default to be the most frequent class in the training set. For each rule $r$ in the rule set, we now count the mis-classifications 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 $R$.

## 5 Example

In order to test our methodology we make use of a database of records taken from the "Student and Staff Telephone Directory" of Purdue University. Each record contains the first, middle, and last, the home address, office address, and telephone number of an individual. Each record contains a system specific key which automatically assigned by the system. This is only done for allowing us to compute the accuracy of the proposed methodology. By using software that controls the generation of errors in the database, we can simulate a variety of real life scenarios. Each replicated and possibly erroneous record maintains its original key. For numerical fields, we compute the Euclidean distance, and for character string fields we compute 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 applying the standardization phase to the records. The records are assumed to be well conditioned because of the database generation process, and for this reason there is no need to apply any other kind of pre-processing to these. For the standardization task, we make use of the Bureau of the Census record matching system [Win99] which includes a module to standardize names, both business and personal, and street addresses.

For sorting the database we can easily resort to the `sort` Unix command which sorts lines of all the named argument files together and writes the results on the standard output. Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line. After the database has been sorted we can easily extract a small portion of contiguous records which will be the sample set for generating the training set. We have used the two first characters from the last and first name as well as the two first characters from the main street name for sorting the database. After that, we can apply the sorted neighborhood approach to the sample set. The size of the window depends among other things on the size of the sample set but we select it to be very small with respect to the size of the sample set. Usually, a window size of 10 is enough. After the application of the sorted neighborhood approach, we end up with the comparison vectors for all the pairs of records falling within a window size of 10 from each other.

The next step in the process is to partition the comparison space into clusters of comparison records that exhibit the same probabilistic behavior for the features used. For this reason we have used AutoClass, a well-known clustering tool developed by NASA Ames Research Center. AutoClass is applicable to observations of objects that can be described by a set of features or properties, without referring to other objects. This allows the user to represent the observations by a data vector corresponding to a fixed attribute set. Attributes are names of measurable or distinguishable properties of the things observed. The data values corresponding to each attribute are thus limited to be either numbers or the elements of a fixed set of attribute specific symbols. AutoClass uses probability models for describing the classes, or else fits probability models to the input data. There are two kinds of models that are supported by AutoClass. The first kind includes probability models which assume that the attributes

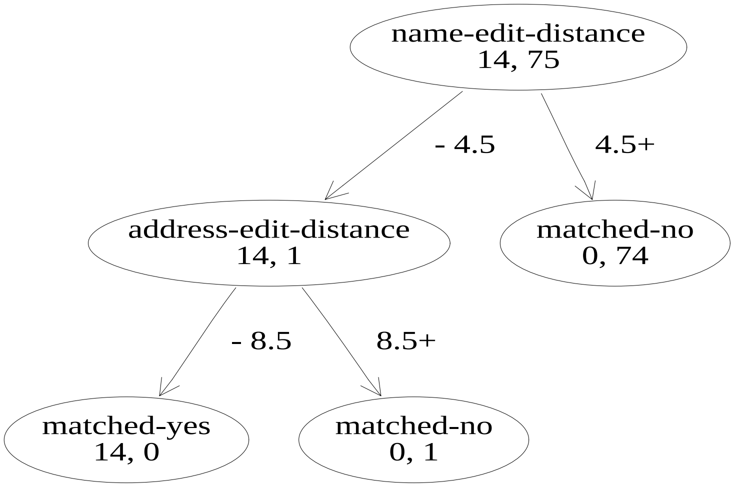

Figure 1: Decision tree generated using 89 cases as a training set.

are conditionally independent given the class, and the second one includes covariant models which can express mutual dependences within the class. The selection of the particular probabilistic model is very important for ensuring the correctness of the results.

The probabilistic model used in this example for all of the components of the comparison vector is the multi normal one. This model implements a likelihood term representing a set of real valued attributes, each with constant measurement error and without missing values, which are conditionally independent of other attributes given the class. This is because of the assumption that an error in a record is independent from any other error. The comparison vector given to AutoClass had 15 components. AutoClass came up with a list of possible models the most probable one containing three “clusters”. This is consistent with the theoretical results of link, non-link and possible link comparison classes. By comparing the actual status of the pairs of records with the classes identified from the clustering tool (recall that for the training set we know the correct matching status of the compared pair of records) we can map each class to the corresponding label. In order to be able to test the predictive accuracy of the induced model, since by having all the records keep the system specific key we know their correct linking status, we use only those linking vectors having as labels the values “matched-yes” and “matched-no”.

For inducing a model for this domain we make use of these two labels along with the corresponding comparison vectors. This information is given to a decision tree inducer for building a model for the identified classes. 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] which also provides a feature subset selection technique for reducing the initial dimension of the comparison vectors. Figure 1 demonstrates the decision tree corresponding to a very small comparison space of 89 comparison vectors.

```
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
```

Figure 2: The rule set produced by translating the information in the induced decision tree.

```
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
```

Figure 3: The rule set produced by simplifying the initial rule set.

By transforming the decision tree patterns from Figure 1 into production rules, we get the rule set displayed in Figure 2. By applying the two-step approach for improving the quality of the rule set, we are left with the reduced rule set displayed in Figure 3 which has exactly the same predictive accuracy as the original rule set at a certain level of statistical significance. The accuracy attained in this example, is 98%. This must be considered with respect to the number of comparison vectors that were assigned to the possible link class and correspond to 4.5% of the total number of comparison vectors used in this case study.

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].

# 6 Conclusions

Data mining techniques (clustering and classification) have been applied to the problem of approximately duplicate record detection. Pruning and certainty factors 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.

# References

[BD83] D. Bitton and D. DeWitt. 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, February 1998.

[EHKU96] A.K. Elmagarmid, B. Horowitz, G. Karabatis, and A. Umar. Issues in Multisystem Integration for Achieving Data Reconciliation and Aspects of Solutions. Technical report, Bellcore Research, September 1996.

[FS69] I. Fellegi and A. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[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.

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

[ME97] A. Monge and C. Elkan. An Efficient Domain Independent Algorithm for Detecting Approximate Duplicate Database Records. In *Proc. of the 1997 SIGMOD Workshop on Research Issues on DMKD*, pages 23–29, 1997.

[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*. Kluwer Academic Publishers, 1997.

[New67] H.B. Newcombe. Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family Histories. *American Journal of Human Genetics*, 19(3):335–339, 1967.

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

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

[Qui87a] J.R. Quinlan. Generating Production Rules From Decision Trees. In *Proc. 10th International Joint Conference on Artificial Intelligence*, pages 304–307, 1987.

[Qui87b] J.R. Quinlan. Simplifying Decision Trees. *Int. J. Man-Machine Studies*, 27:221–234, 1987.

[Qui96] J.R. Quinlan. Learning Decision Tree Classifiers. *ACM Computing Surveys*, 28(1):71–72, 1996.

[Win99] W.E. Winkler. *Record Linkage Software*. U.S. Bureau of the Census, 1999.

[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.