On the Accuracy and Completeness of the Record Matching Process*

Vassilios S. Verykios[1], Mohamed G. Elfeky[2], Ahmed K. Elmagarmid[2], Munir Cochinwala[3] and Sid Dalal[3]

[1]*College of Information Science and Technology, Drexel University*
[2]*Department of Computer Sciences, Purdue University*
[3]*Applied Research Division, Telcordia Technologies*

Abstract. Record matching or linking is one of the phases of the data quality improvement process, in which, records from different sources, are cleansed and integrated in a centralized data store to be used for various purposes. Both, earlier and recent studies in data quality and record linkage focus on various statistical models, which make strong assumptions on the probabilities of attribute errors. In this study, we evaluate different models for record linkage, which are built based on data only. We use a program that generates data with known error distributions and we train classification models, which we use to estimate the accuracy and the completeness of the record linking process. The results indicate that the automated learning techniques are adequate for this process and that both their accuracy and their completeness are comparable to the accuracy and the completeness of other, mostly manual, processes.

1. Introduction

The enterprise data is created, used and shared by a corporation in conducting business. This is a critical business asset that must be designed, analyzed, and managed with data quality as a guiding principle and not as an afterthought [2][7]. Poor data quality, that results from missing customer information, wrong address information, etc., undermines the customer satisfaction, leads to high and unnecessary costs and more importantly impacts decision making [20][21][17]. A number of reasons is responsible for bad data quality including: (a) multiple sources of data, (b) incompatible data, (c) data from multiple level of granularity, (d) redundant data, (e) corrupted and noisy data, and (f) missing attribute values.

Data quality is achieved in three stages: the first one is based on the cleansing, or scrubbing of data, the second one on the matching or the linking of the records and the third one on the consolidation or the integration of the cleansed information with other internal or external sources. In the first stage, the data is parsed, corrected, standardized and enhanced so that the linking phase will be as accurate as possible. In the second

* This work was supported in part by Telcordia Technologies, IBM, HP, NCR, and CERIAS.

phase, comparisons are made within and across the various data sources in order to locate similar information. Finally, the matching information is integrated and placed into a warehouse, a data mart or another data storage area.

In this study we are primarily interested in the record matching or the linking phase, which is a part of the entire data quality process. The record matching phase determines whether two records or rows, either of the same type or not, represent data on the same object. This phase involves many value judgments and requires sophisticated algorithms and software tools. Customer matching and retail house holding are two examples on which the record matching phase can be applied. In the first application, we try to identify if a customer is buying for a second, third, or a multiple time, a product from the same supplier. In the second application, we want to find different groups of people who comprise a family, each member of which is a customer.

Record matching or linking is very similar in practice to the duplicate record detection, in which all records in a file or database are found that contain exactly, or approximately, the same data in one or more fields. In the following discussion linking and matching will be used interchangeably although there is a subtle difference between these two terms when we refer to this process. In details, *matching* refers to the actual relationship between a pair of records, while *linking* refers to the decision taken by a matching algorithm with respect to the linking status of a pair of records. Matching allows for identification of similar data within and across the data sources. By using cleansed information and match standards, we can eliminate the duplicate representations and integrate all the information about each individual customer or about an entire household. This will allow to examine thoroughly each customer or household, to generate accurate data about them, to enhance response rates or marketing promotions, to identify trends and patterns and finally to accurately target new prospects.

In this paper, we are considering the record linking process as a classification task and we are evaluating two different approaches for developing classification models for matching records. In Section 2 we will provide some background related to the record matching and the related technology. In Section 3 we will describe the formulation of the problem and the learning approaches that we use for evaluation. Section 4 presents the experimental system that was built, and Section 5 presents the experiments that we conducted along with the results. Finally, in Section 6 we summarize our observations, and we point out to some extensions of this work.

2. Background

Record matching or linking is the process of identifying records, in an electronic file or database, that refer to the same real world entity or object. The decision, as to the matching status of a pair of records, is based on the comparison of common characteristics between the particular pair of records. These common characteristics are related to the similarities in the schema of the corresponding records. For example, a customer table can have two different schema representations in two databases, both storing customer data. The first table may store information from the service department

and the second one, may store information from the billing department. Despite the differences in the representation of the two tables, overlapping information (i.e., name, address, sex, marital status, etc.) can always be present, and this information can be used for the identification of matches of records from different databases that refer to the same customer.

The record linking or matching process consists of two basic steps. The first one is the *searching step* and the second one is the *matching step*. In the searching step, the record linking process explores the space of record pairs in order to identify records that are prospective matches. We expect that only a very small percentage of the records that exist in a file or database, will actually match. In the matching step, the record linking process compares the records, which were identified as prospective matches during the searching phase, and assigns to them one of the following labels: *link*, *non-link*. The result of the comparison between any two records depends on the nature of the common characteristics in the two databases.

Whenever there is not enough evidence to choose one of the decisions over the other, there is a third decision that lies in between the first two ones. In that case, the record pair is assigned the *possible link* status. All the record pairs, that have been assigned the possible link status, should be manually reviewed in order for a final decision to be reached for that pair. It is obvious that a matching algorithm that assigns too many pairs that need manual inspection, should be avoided, because this will introduce a high overhead to the matching process. For this reason, by using either some cost metrics or some levels of acceptable error in the record linking process, we can explicitly control how many of the record pairs can be assigned the possible link status. In the following two subsections, we summarize the state of the art in both the searching and the matching process.

2.1 Searching Process

The searching process must be intelligent enough in order to exclude from comparison, record pairs that completely disagree with each other (not prospective matches). This is quite reasonable based on the fact that the matching process has a very high application cost and must be applied economically. Often times, we need to make a compromise between the number of record pairs, that are compared, and the accuracy of the matching process. For example, let us assume that we have developed a matching algorithm, which is error free. It is clear that such an algorithm is very difficult to be built and we expect that its complexity should be high enough. Given such a matching algorithm, the only way to be sure that we have found all the matching record pairs is to apply an exhaustive pair-wise comparison of all records in both data sets. The complexity of such an approach is quadratic in the number of records in the database. If we also assume that a very small percentage of the pairs of records that will be undergone a comparison, will be actually a match, we will end up with an extremely high process overhead. The solution to this problem is to identify only those record pairs (prospective matches), which have high probability of matching, leaving not inspected those pairs that look very different.

Several techniques have been developed in the past searching the space of record pairs. The first one, which was presented early on in a paper by Newcombe [15] is called, *blocking*. This technique partitions the entire set of records into segments, which are called *blocks*. Each block contains records that have the same value for a subset or part of the common characteristics. These characteristics are known as *blocking variables*. The idea behind blocking is to compare records from the two data sets that belong to blocks, which agree on the blocking variables. This decreases the overall complexity of matching but it may also exclude some of the matching records from being inspected. This approach is very similar to what is called *clustering* in [10].

The second approach consists of two steps and is similar to the approach that is used for duplicate detection [3]. During the first step the database is sorted. In the second step, the searching process looks for records that are prospective matches but only in small regions in the database. This approach leaves the searching process with the task of comparing only those pairs of records that lie next to each other in the final ordering of the database. The subset of the characteristics, which is used for sorting the data sets, is known as the *sorting key*. The role of the sorting key is very similar to that of the blocking variable. In this approach, it is clear that the data sets should be merged into one set, before the sorting starts.

A very similar approach to the sorting technique, which increases the completeness of the matching process, is called *windowing* or else the *sorted-neighborhood approach* and it was proposed by Hernadez et. al in [10]. In this approach, the data sets are merged and then sorted, like in the sorting approach. The only difference is that instead of comparing records that lie next to each other, if they only agree in the characteristics selected for sorting, the compared records needs only to fall within a constant size window from each other. If the window size is very large this process is equivalent to the exhaustive search method. If now the window size is small, the process is very efficient and gives better results than the sorting method.

Because of the errors that exist in the data sets that are compared, it is very common that the information selected for blocking or sorting the data sets contains errors. If that happens, we expect that some records are clustered far away from those records with which they should be compared to. For this reason, a *multi-pass approach*, proposed in [10], can be used. In this approach, a number of different blocking variables, or sorting keys, can be used for clustering the records in different ways. The database is then scanned as many times as the number of the different keys. The results from independent passes are combined to give the final set of matching records. The same group of researchers has also implemented an extension of the multi-pass approach in which the transitive closure of the results of independent passes is computed. A similar approach, that has been proposed independently by Monge et. al. in [14] makes use of an algorithmic technique for identifying the connected components of a graph. By considering each record cluster as a connected component, this process can be effectively used to identify the records that belong to the same cluster. Both groups of researchers presented very similar results regarding the accuracy and the cost of the searching process.

2.2 Matching Process

The most important issue in the matching process, is to take a correct decision about the matching status of a pair of records by looking at their characteristics. A comparison between the common characteristics is required before making a decision based on the agreements or disagreements of these characteristics. The comparison of the characteristics depends heavily on the nature of the characteristics. As these characteristics include strings of characters, the string comparison algorithms play an important role in the record matching process. Some kinds of character strings, like names, exhibit a certain error behavior, and for this reason, some coding schemes have been devised. These coding schemes, mostly phonetic, like the Soundex code [15], can be used to extract some components that are very prone to errors from the name string, so as the string matching process will be more robust and reliable. String matching algorithms, like the edit distance or the Jaro algorithm [22], or the n-grams [11], have been used in different research and experimental record matching models or systems and they gave very promising and accurate results.

The simplest way to solve the record matching problem is to consider it as an extension of the string matching problem. In this way, a record linkage rule can be built, by using a string matching algorithm. This type of algorithm considers the entire database record as a string with or without blank characters and decides upon the matching status based on the distance of the strings corresponding to the records under comparison. This approach has been called *field matching* in [14] and its performance depends primarily on the selection of a very smart string matching algorithm that accounts for different types of errors. We believe that such an approach is not powerful enough to provide a general solution for the record matching problem, unless strong assumptions on the distributions and the types of errors can be given beforehand.

The first general solution to the matching problem was probabilistic in nature. An optimal linkage rule was first proposed by Newcombe [15]. Later on, a record linkage theory was strictly formulated by Fellegi and Sunter [9] based on the matching and non-matching probabilities of the record comparisons. In other words, they observed that a certain comparison pattern has a different frequency of occurrence among records that do match than among records that do not match. By dividing these probabilities for each pattern and by sorting the ratios, they generated a linear ordering of these agreement and disagreement patterns. The next problem that they came along, was the identification of appropriate threshold values for these ratios, in such a way that the linkage rule could designate a pair as a link or non-link just by looking at the ratio value. Fellegi and Sunter computed these thresholds by using the expected probabilities of errors, and they bounded their values, by the error levels desired by the user. The optimality of the linkage rule proposed by Fellegi and Sunter relies on the fact that it minimizes the number of record pairs that are assigned a possible link status. The reasoning behind this selection is that pairs, designated by the linkage rule as possible links, need to be manually inspected. But, as we mentioned earlier, manual inspection costs a lot and should be avoided, if possible.

The exploitation of an *equational theory* [10], which could be developed based on human expertise, was considered as an alternative solution. This approach is known as the knowledge acquisition approach of knowledge engineering. An equational theory is actually a set of rules that captures the semantics of the record linkage process. This set of rules is built based on an interview of a domain expert with a knowledge engineer. Although the researchers presented good results, we should take into consideration that such techniques have certain mostly computational limitations (i.e., knowledge acquisition bottleneck) and should not be used occasionally.

The problem with the probabilistic approach, which was presented earlier, is that the matching and non-matching probabilities of the comparison patterns should be considered known a-priori. Although this is a legitimate assumption, usually, such probabilities cannot be estimated beforehand. A different model for record linkage, based on machine learning and statistical techniques, was developed by Cochinwala et. al. in [4]. The authors consider a very small sample of data, which is semi-automatically processed, in order to build a training set, for inducing a model for matching records. In their paper, the induced model, was a decision tree. The decision tree by itself, does feature selection which decreases by orders of magnitude the cost of matching. In order to further decrease the complexity of taking a decision by using the induced model, the authors use a model selection technique for pruning the initially induced model by using complexity costs. A fully automated methodology to build a similar model, has been proposed by Verykios et. al. in [18][19].

A very similar approach with the one proposed by Verykios et. al. and Cochinwala et. al., is the approach proposed by Debabrata et. al., for sequential record matching [5]. The authors propose to build a model incrementally. By doing so, they avoid the expensive operation of transferring all the data in a local store at once. They also propose to decide, in an on-line fashion, on the set of attributes that are necessary, for taking a correct decision. Debabrata et. al. in [6], present a different model for record linkage based on distance metrics.

3. Methodological Framework

Our main goal of this study is to build on top of previous studies, which were focused on generating a decision model for record linkage by looking at the data itself without imposing any assumptions about the probability distribution of the expected errors in the data [4][19]. In particular we try to evaluate different searching and matching methodologies with respect to the *accuracy* and the *completeness*. The completeness of the process is related mostly to the searching phase (Section 5.2), while the accuracy of the process is related mostly to the matching phase (Section 5.1).

The *accuracy* of the record linkage process is defined as the number of record pairs, which are brought together for comparison, and are assigned the link status by the record linkage decision model, over the actual matching pairs of records that are brought together for comparison. As we will see later on, the accuracy of the decision model is

equivalent with the predictive accuracy of the classification model that is used for decision purposes.

The *completeness* of the general process has two aspects. In the first one, we assume that the record linkage decision model, which decides upon the linking status of a pair of records, is a perfect one. We refer to this type of completeness as the *searching completeness*. Based on this assumption, the searching completeness can be defined as the percentage of the matching records that are brought for comparison, over the total number of matching records. Records that are not brought together for comparison are implicitly assumed as not matching. The second aspect is related to the completeness of the entire process, assuming an imperfect record linkage decision model. We refer to this type of completeness as the *linking completeness*. Under this assumption, the linking completeness is the product of the searching completeness and the accuracy of the matching phase.

There are certain indicators that can be used to estimate the relative merit of the different fields or attributes in a record, for clustering or sorting the records in the database. These indicators are closely related to the completeness of the record linkage process and they are briefly described in this paragraph. There are three numerical tests used for this reason. The first one is the *coefficient of specificity*, the second one is the *discriminating power* and the third one is the so-called *merit ratio*. The coefficient of specificity is the fineness with which a file will be divided by a particular kind of identifying information. Unlike the coefficient of specificity, which gets smaller as a database becomes more finely divided, the discriminating power increases with the extent of the subdivision. The discriminating power is defined as the logarithm of the inverse of the coefficient of specificity. Finally, the merit of any particular kind of identifying information for sorting the file, may be taken as the ratio of the discriminating power to the likelihood of discrepancy or inconsistency of such information. This is the so-called merit ratio.

In this study, we aim at building a record linkage decision model automatically, by using inductive learning techniques, which have been implemented in public domain software. In particular, we focus on comparing the predictive accuracy and completeness of decision models which were built by using two of the common learning paradigms in AI, the decision tree and the instance-based methods, both of which are also well known in statistics. These methods share an approach to learning that is based on exploiting regularities among observations, so that predictions are made on the basis of similar previously encountered situations. The methods differ, however, in the way that similarity is expressed; trees make important shared properties explicit, whereas instance-based approaches equate (dis)similarity with some measure of distance. An example of a decision tree, which we generated in these experiments, is shown in Figure 1.

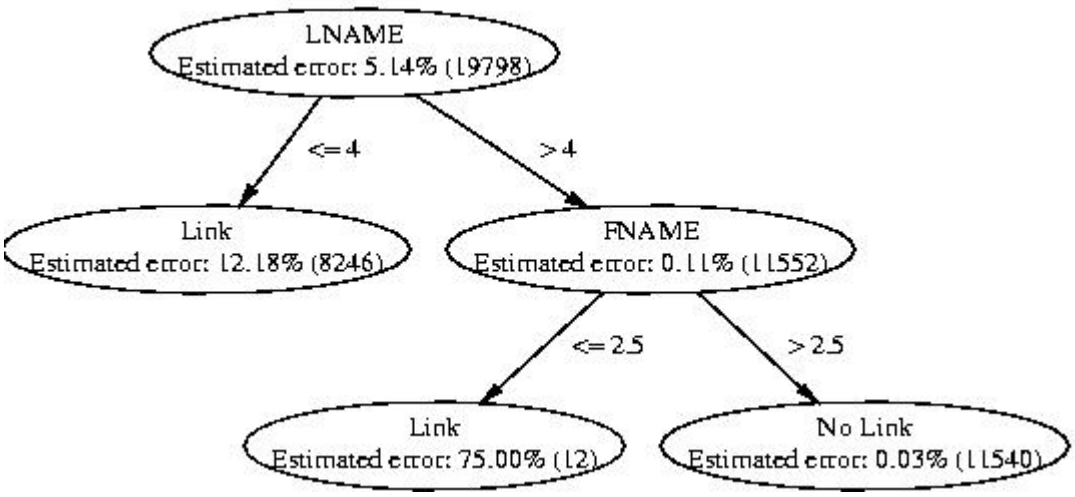

Figure 1: Example of a Decision Tree.

The specific learning algorithms used in the experiments, are implemented in the MLC++ library [12]. The first algorithm is a decision tree inducer and it is described in [16], while the second one is a instance-based learning algorithm which is described in [1]. Decision tree and instance-based methods both represent each instance using a collection $\{A_1, A_2,\ldots,A_x\}$of properties or attributes. Attributes are grouped into two broad categories: continuous attributes have real or integer values, whereas discrete attributes have unordered nominal values drawn from a usually small set of possibilities defined for that attribute. Each instance also belongs to one of a fixed set of mutually exclusive classes $c_1, c_2,\ldots,c_k$. Both families of methods use a training set of classified instances to develop a mapping from attribute values to classes; this mapping can then be used to predict the class of a new instance from its attribute values.

The validation of the induced models in order to measure their predictive accuracy, was another important aspect in our experiments. There are various methods that can be used for estimating the predictive accuracy of a model such as re-substitution estimate, holdout, cross validation, leave one out, and bootstrapping method. More information about these methods can be found in [12]. In our experiments, we made use of the holdout method. This method randomly partitions the data set into two mutually disjoint sets: the training set and the test or the holdout set. The induction algorithm builds the model based on the cases included in the training set and then evaluates the performance of the induced model by using the data in the test set. The predictive accuracy is computed as the number of correctly predicted instances in the test set, over the total number of instances in the test set.

The effective calculation of the completeness of the record matching process was yet another issue that had to be investigated in the experiments that we conducted. In Section 5.2, we further elaborate on this subject matter.

4. Experimental System Overview

In order to evaluate the performance of the various schemes that we already described, we used two public domain software tools. The first tool was a database generator, which was used for automatically generating the databases used in the experiments. The database generator, which is described in [10], provides us with a large number of parameters such as the size of the database, the percentage of duplicate records in the database and the amount of error to be introduced in the duplicated records in any of the attribute fields. Each generated record consists of the following fields: social security number, first name, middle initial, last name, address, apartment number, city, state, and zip code. We must point out that based on the settings that we select for the generator, not all of the field values are present in each record.

The experimental system that we developed, takes as input two files: the first one contains the training set, and the other one, the test set. Each one of these files, contain a certain number of comparison vectors. Every comparison vector has a fixed size, which is equal to the number of field comparisons between a pair of database records generated by the database generator. By using the database generator, the two files or databases are generated. The first one is used for generating the training set and is usually small compared to the other one, which is used for generating the test set.

The system also, makes use of a configuration file. The configuration file, that describes the current experiment, contains three parts. The first part specifies the sizes of the database files to be generated. Since we may use different sizes of files for training and for testing, the first two parameters specify the training set size and the test set sizes. By using the terminology in [10], we define as a *cluster*, the group of records that are actually duplicates. The *cluster ratio* is the ratio between the number of records and the number of clusters generated in the database. In order to ensure the consistency of the results for each experiment, the cluster ratio should be constant for the databases generated. Yet it may be required that the ratio for the training set is different from that of the test set. Therefore, the other two parameters in this part are the training set cluster ratio and test set cluster ratio.

After reading the first part of the configuration file, the system runs the database generator system to generate the training and the test databases according to the sizes and the cluster ratios specified. It combines each training database with each test database to form one case in the current experiment. The remaining two parts in the configuration file, specify how to generate the comparison vectors for each case. These two parts contain values for the same set of parameters, one for the training set and the other for the test set. Some of these parameters are documented below. The two main issues that should be taken into account before generating the comparison vectors from the database records, are the sorting and the selection of the records, which are to be compared. For this reason, we partition the remaining part of the configuration file into two groups.

The first group addresses the issue of sorting. The *sorting key* parameter, specifies the set of fields by which the records will be sorted. The *characters used* parameter, specifies how many characters to be used from each field for sorting purposes. The Soundex code

is described in [15] and it is another possibility as far as the sorting of the records is concerned. The *Soundex* parameter in the configuration file, is a binary parameter that specify whether the records will be sorted based on the Soundex code or not. The second group of parameters accounts for the preparation of the sets of comparison records. The blocking [15] and the sorted-neighborhood [10] approach, are two possible options for this task. The blocking parameter determines whether blocking will be used or not. If the sorted-neighborhood method is to be used, the blocking parameter is set to zero and an additional parameter, which specifies the window size, is deployed.

After reading these two sets of parameters for the training set and for the test set, the system generates two groups of record pairs one for training and one for testing. Then each group is transformed into a group of comparison vectors. The calculation of the comparison vectors, between any pair of records, is done by measuring the edit distance [13] between the string fields (i.e., first name, last name, etc.) and the Hamming distance of the numeric fields (i.e., SSN, Zip Code, street number, etc.). The exact actual matching status for each comparison vector is known a-priori for both the training and the test set. Therefore, each comparison vector is associated with a label according to its actual status. This can happen because the database generator creates one more field in every record, which is called *cluster identifier*. This identifier indicates the cluster that every record belongs to. A cluster represents a set of records that correspond to the same real-world entity.

5. Experiments and Results

In the following subsections, we present the experiments that were conducted, and the results that we observed, along with various comments on them. In Section 5.1 we present experiments for the evaluation of the accuracy of the induction methods, while in Section 5.2, we present experiments related to the completeness of the matching process.

5.1 Accuracy

As we have already described, we used two induction algorithms: ID3, a decision tree inducer, and IBL, an instance based classifier. For running these algorithms, we made use of the programs provided by MLC++. The parameters for these programs are the default parameters supported by the library. The only selection that we made was the number of neighbors to be used in the IBL algorithm. In this study, we select three neighbors, while we have experimented with other values as well, and we had similar results.

The parameters in the configuration file are summarized below. For training purposes, we generated a database of 50 records by using the database generator tool that is described previously. We also generated six different databases for testing purposes. The sizes in records of these databases are: 500, 1,000, 2,500, 5,000, 10,000, and 20,000 records. For each combination of training and testing set, we set the cluster to database ratio or else the approximate number of records per cluster to the following values: 0.5, 0.25, 0.2 and 0.1. The values of the ratio correspond to the following numbers or records per cluster: 2, 4, 5 and 10. The ratio was the same for both training and testing purposes.

Before we sort the files, we transformed all characters in the database to uppercase letters. As a sorting key, for both the training and the test set, we selected the last name of the record and in particular the Soundex code of the last name. In all of our experiments related to the accuracy, we used the sorted-neighborhood approach for generating the training set. For the sorted-neighborhood approach we made use of the following window sizes: 2, 4, 5, and 10. The idea behind this selection was to use a window size that is equal to the number of records in the clusters generated by the database generator so that we can spend the minimum computational effort for learning the *link* concept.

After making these selections, we performed two sets of experiments. The main goal was to determine which one of the two selected classifiers performs better with respect to the accuracy, and how this accuracy varies with respect to two alternatives in the test set generation process: (a) by using the blocking approach, and (b) by using the sorted-neighborhood approach. For the first set of experiments we used the blocking approach and we selected the entire sorting key, which was the Soundex code of the last name, as the blocking variable. The miss-classification rates for the set of four combinations of *cluster ratio* and *window size* are shown in Figure 2.

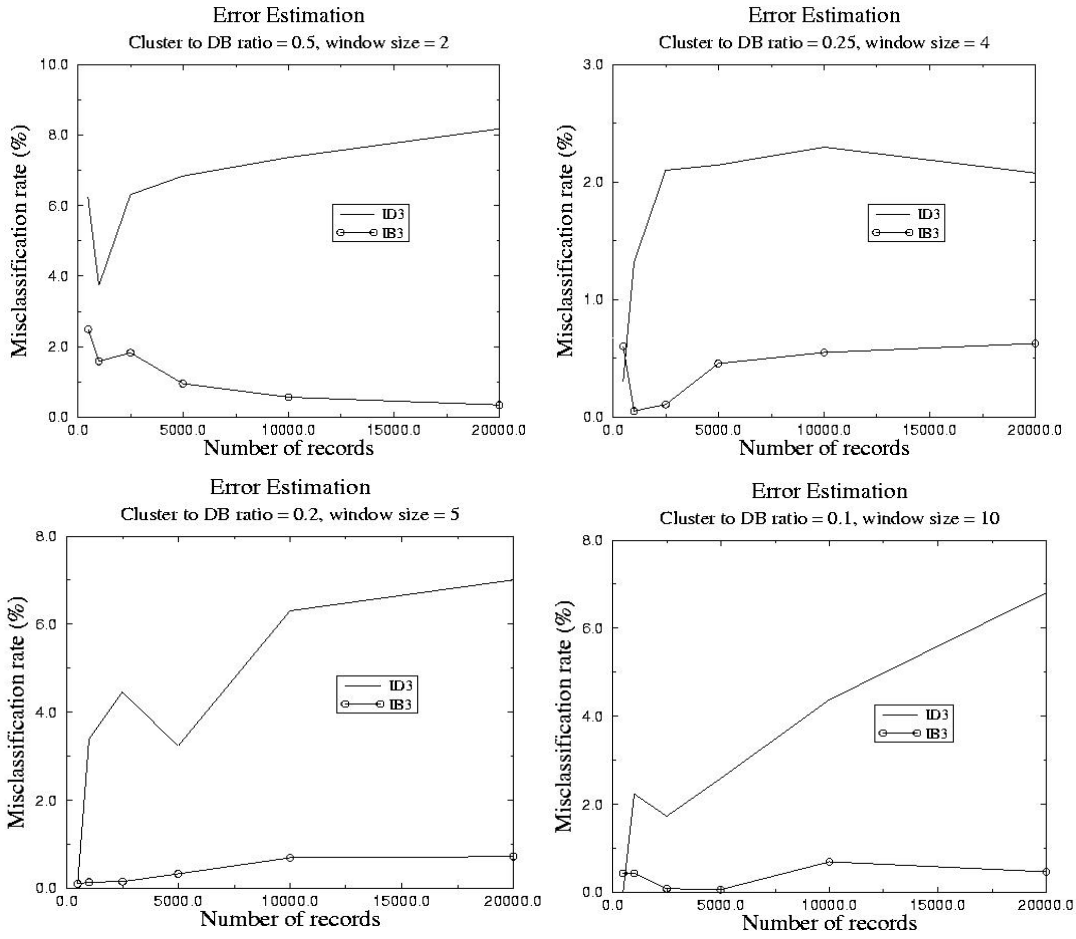

Figure 2: Misclassification rates with blocking in the test set.

For the second set of experiments and for the generation of the test set, we selected to use the sorted-neighborhood approach, instead of the blocking approach. In order to be consistent with the selections of parameters that we made for the training set, we also selected the following values for the window sizes in the test set generation process: 2, 4, 5, and 10. These values are exactly the same with the ones that we used for the training set generation process. The results of this set of experiments are shown in Figure 3.

The graphs in Figure 2 and Figure 3 demonstrate that IB3 (the IBL algorithm with three neighbors) outperforms ID3 in almost all of the experiments that we conducted. This superiority is irrelevant to whether we use a blocking or a sorted-neighborhood approach for the generation of the test set. The only disadvantage of the instance based algorithm is that it always takes a considerably larger amount for its execution.

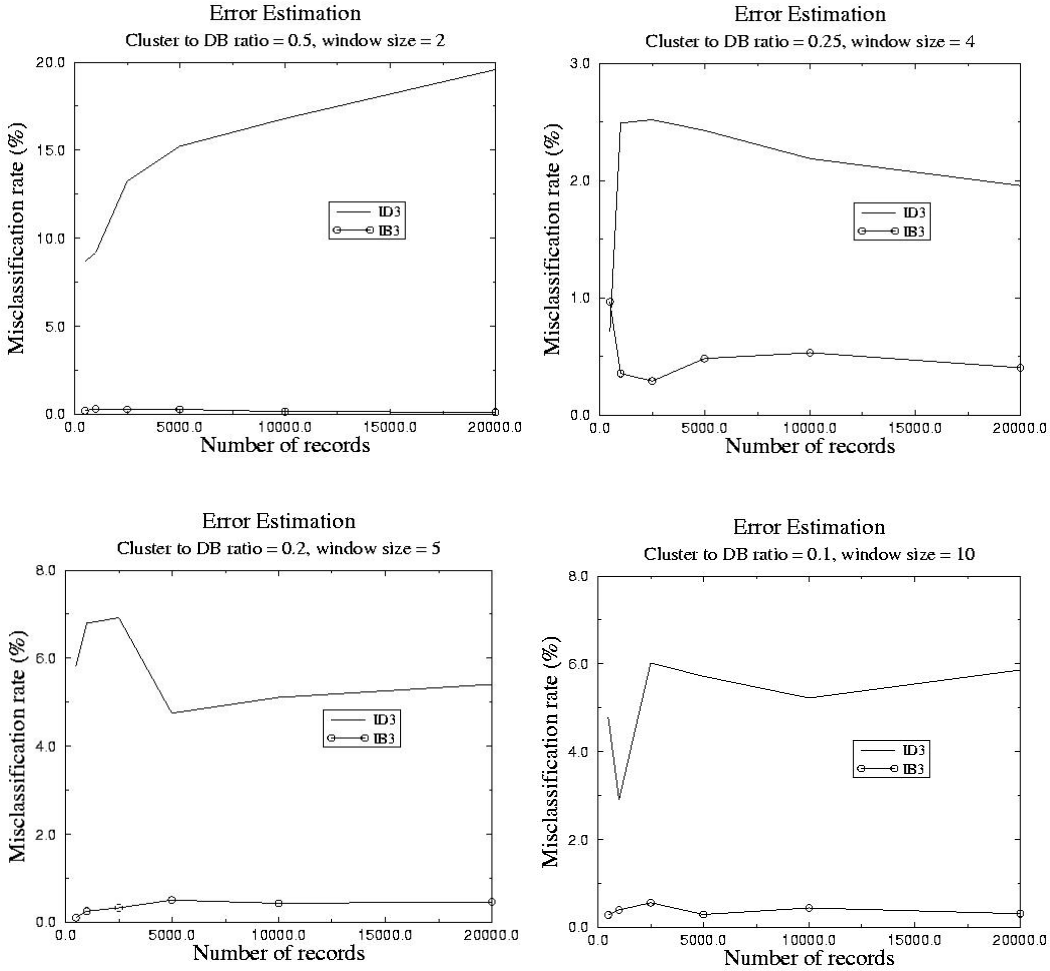

Figure 3: Misclassification rates without blocking in the test set.

5.2 Completeness

In the experiments that we describe below, we restrict ourselves to considering only the linking completeness but not the *searching completeness*. In order to be able to argue

about the completeness, we need to be able to compute the number of the exact links that are discovered in the record matching process. Hence, we need a way to reconstruct the clusters of duplicate records based on the model built by the rule induction technique used. Therefore, we built a subsystem that performs this operation with the help of a categorization utility of MLC++. When this utility runs, it produces labels for the comparison vectors based on the induction model built before. Our subsystem reads these labels and tries to reconstruct the clusters of duplicate records. Specifically, for each comparison vector that is labeled *link* by the categorization utility, our subsystem inserts the corresponding database records into the same cluster. If we repeat this process for all the comparison vectors, we end up with a group of clusters of the duplicate records discovered by the model. If we compare this group of clusters to the original one (known from the database generation step), we can calculate the completeness as follows. We generate two sets of duplicate record pairs, the first from the original clusters and the second from the discovered ones. The completeness is the size of the intersection between the two sets divided by the size of the original set. This value represents how many duplicate record links are discovered out of the original previously known ones.

The first experiment for completeness uses the Soundex code of the last name as the sorting key for both the training and the test sets. It uses the sorted-neighborhood method for generating the comparison vectors fixing the window size for the training set to 10. The results of this experiment are shown in Figure 4(a). The completeness of the model is increased with the increase of the window size value of the test set over the same test set size. This is obvious since the increase of the window size of the test set increases the number of the generated comparison vectors and consequently increases the probability of detecting the duplicate records. Moreover, the results show that the completeness is decreased with the increase of the test set size over the same test window size.

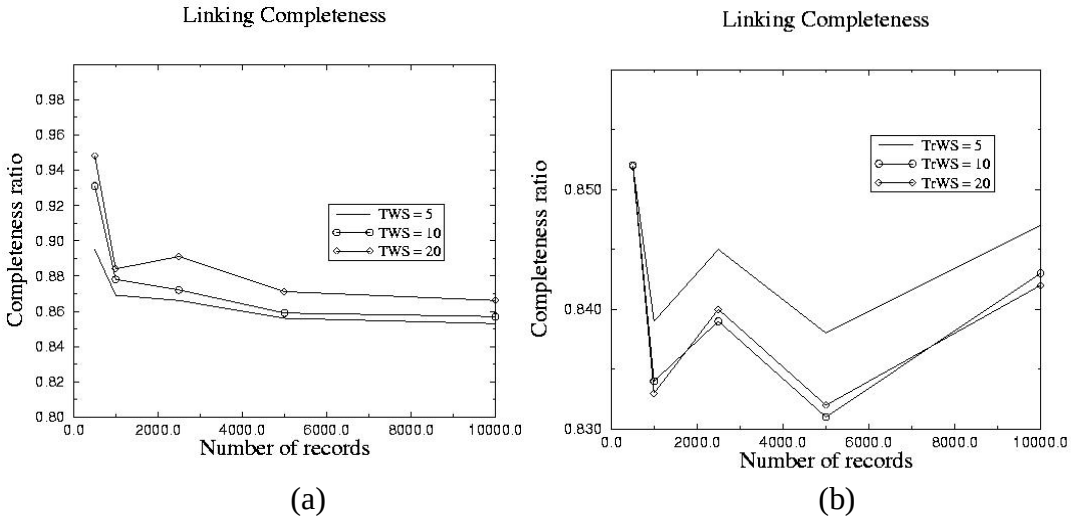

Figure 4: Estimates of the linking completeness.

In the second experiment we also use the Soundex code of the last name as the sorting key for both the training and the test sets. In order to generate the comparison vectors, we use blocking for the test set, and the sorted-neighborhood method for the training set. The results are shown in Figure 4(b). The first observation is that the completeness values in the first experiment are somehow larger than the values in this experiment. This means that the sorted-neighborhood method is better than blocking although it may generate more comparison vectors and requires more time. Our second observation is that there is no noticeable change with the change of the window size of the training set. Moreover, the increase of the window size of the training set, after a certain value, will not affect the model induced and so will not affect the completeness value for the same test set size. We must point out that the use of the blocking approach for the generation of the comparison vectors of the training set is not worthwhile and does not generate a good model. Hence, we conclude our completeness experimental results as follows: (a) the sorted-neighborhood method is better than the blocking method, (b) the increase of the window size of the test set results in the increase of the completeness value, and (c) the increase of the window size of the training set has no effect. We must notice that all these experimental results are due only to ID3 model. The reason for not presenting results for the completeness of IB3 is because this inducer does not support a persistence model according to the MLC++ library that we used. In our future work, we will build a categorization utility for this system and we will conduct similar experiments.

6. Conclusions

The data quality problems are commonplace in the enterprise level, as organizations downsize, merge and consolidate. An important issue related to the quality of the data, is how an integrated view of the data can be built. It is often the case, that different legacy systems, storing disjoint or overlapping information, have different levels of quality in the stored data. An automatic way to identify common information among these systems is of great importance before we can integrate these data.

In this paper, we have reported results on the accuracy and the completeness of various schemes, which deal with the problem of matching common information. Our experimental system can be fully exploited to support such studies. The preliminary results shown in this paper, indicate that very accurate and complete models can be built by using commercial off the shelf software tools along with real data, without making any strong assumptions about the probabilities of errors in the data.

There is a lot of experimentation that needs to be done, before we are able to make any general assertions about the tool, model or parameter selection scheme for the record matching process. We are in the process of conducting these experiments, while we are extending our experimental system to support fully automated approaches for matching records.

References

[1] D. W. Aha, Tolerating Noisy, *Irrelevant and Novel Attributes in Instance-Based Learning Algorithms*, International Journal of Man-Machine Studies **36**, (1992), no. 1, 267-287.

[2] Amjad Umar, George Karabatis, Linda Ness, Bruce Horowitz, and Ahmed Elmagarmid, *Enterprise Data quality: A Pragmatic Approach*, Information Systems Frontiers **1** (2000), no 3, 279-301.

[3] D. Bitton and D. DeWitt, *Duplicate Record Elimination in Large Data Files*, ACM Transactions on Database Systems **8** (1983), no. 2, 255-265.

[4] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha, *Efficient Data Reconciliation*, Bellcore Report, February 1998.

[5] Debabrata Dey and Vijay Mookerjee, *A Sequential Technique for Efficient Record Linkage*, submitted to Operational Research Journal, 2000.

[6] Debabrata Dey, Sumit Sarkar, and Pradudha De, Entity Matching in Heterogeneous Databases: A Distance Based Decision Model, Proceedings of the 31st Hawaii International Conference on System Sciences, 1998.

[7] D. Georgakopoulos and G. Karabatis and S. Gantimahapatruni, *Specification and Management of Interdependent Data in Operational Systems and Data Warehouses* Distributed and Parallel Databases **5** (1997), no. 2, 121-166.

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

[9] Fellegi and A. Sunter, *A Theory for Record Linkage*, Journal of the American Statistical Association **64** (1969), no. 328, 1183-1210.

[10] 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** (1998), no. 2.

[11] Jeremy A. Hylton, *Identifying and Merging Related Bibliographic Records*, Master's Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

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

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

[14] A. Monge and C. Elkan, *An Efficient Domain Independent Algorithm for Detecting Approximate Duplicate Database Records*, Proceedings of the 1997 SIGMOD Workshop on Research Issues on DMKD, 1997, pp. 23-29.

[15] H.B. Newcombe, *Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family Histories*, American Journal of Human Genetics **19** (1967), no. 3, 335-339.

[16] J.R. Quinlan, *Induction of Decision Trees*, Machine Learning 1 (1986), no. 1, 81-106.

[17] T.C. Redman, *Data Quality for the Information Age*, Artech House Publishers, 1996.

[18] V.S. Verykios, A.K. Elmagarmid, and E.N. Houstis, *Record Matching to Improve Data Quality*, Technical Report TR-99-005, Department of Computer Sciences, Purdue University, 1999.

[19] V.S. Verykios, A.K. Elmagarmid, and E.N. Houstis, *Automating the Approximate Record Matching Process*, Information Sciences **126** (2000) 83-98.

[20] Y. Wand and R.Y. Wang, *Anchoring Data Quality Dimensions in Ontological Foundations*, Communications of the ACM **39** (1996), no. 11, 86-95.

[21] 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** (1995), no. 4, 623-640.

[22] W.E. Winkler, *Advanced Methods for Record Linking*, Proceedings of the Section on Survey Research Methods (American Statistical Association), pp. 467-472, 1994.