2001

Record Matching: Past, Present and Future

M. Cochinwala

S. Dalal

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

V. S. Verykios

Report Number:
01-013

RECORD MATCHING: PAST, PRESENT AND FUTURE

M. Cochinwala
S. Dalal
A.K. Elmagarmid
V.S. Verykios

CSD TR #01-013
July 2001

Record Matching: Past, Present and Future*

M. Cochinwala[1], S. Dalal[1], A.K. Elmagarmid[2], V.S. Verykios[3]

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

March 19, 2001

Abstract

Data quality often manifests itself as inconsistencies between systems or inconsistencies with reality. The latter can be explained as a mismatch between real world objects and the way they are represented in databases. The former, as it is exemplified by data heterogeneity - semantic and structural - replicated data, and data integrity problems, is the main cause of duplicate records. Often the same real world entity is represented by two or more records. Many times, duplicate records do not share a common key and/or they contain errors that makes matching them a difficult problem. This specific problem is the subject of this expository study. This study relies on a thorough analysis of the literature, on an intimate knowledge of an application and on many years of working on data quality with sensitive telecommunication systems. This paper also presents a taxonomy for record matching algorithms and proposed solutions. This paper is limited to record matching and addresses the general problem of data quality only in passing. This should be and has been the topic of many other survey papers.

---

*Content Indicators: Information Systems, Computing Methodologies, Models and Principles, Software, Data

Contents

1 Introduction

Databases play an important role in today's IT based economy therefore their quality carry a potentially high price on the systems that depend on these databases in order to stay operational. In an error-free system with perfectly clean data, the construction of a global view of the data consists of linking – in relational terms, joining – two or more tables on their key fields. Unfortunately, very often these data are neither carefully controlled for quality nor necessarily defined commonly across different data sources. Data quality thus, is compromised due to problems in design and improper data handling. Lack of normalization, or de-normalization, missing integrity constraints are some aspects of design-based data quality issues. Wrong or missing data, un-controlled data duplication, or other related logical errors are some forms of improper data handling. As a result, the creation of such a global data view resorts to "fuzzy" or approximate joins.

To make things worse, global applications require information from several databases in order to run. If these databases are independently managed, the same data is likely to be represented differently in these databases. Not only the values, but the semantics, the underlying assumptions and the integrity rules may differ as well. For this reason, databases can accrue a wide range of inaccuracies and inconsistencies, such as misspelled names, inverted text, missing fields and outdated area or ZIP codes. These problems are related to the incompatible representation of the data and they are known as *data heterogeneity* problems or else as *instance identification problems* [5, 42, 37, 23, 4, 24, 1]. *Data cleaning*, or *data scrubbing*, are two terms related to the process of resolving such problems in the data [14]. Organizations usually become aware of these problems while they implement a data warehouse as they start combining data from different legacy systems. During this process, they may discover formatting problems, definition problems and other types of structural problems.

We distinguish between two types of data heterogeneity: *structural* and *semantic*. The structural heterogeneity occurs when the attributes are defined differently in different databases. The semantic heterogeneity occurs when similarly structured attributes take on different semantics and values in different databases. Data inconsistencies result in organizations being unable to quickly assess the state of their business. In this paper, we survey various techniques which have been developed for addressing a problem related to the semantic heterogeneity, and in particular, to value inconsistencies. This problem has been known for more than five decades as the *record linkage* or the *record matching* problem [31, 30, 32, 38, 15, 33, 43]. The goal in the record matching problem is to identify records in the same or different data sources/databases that refer to the same real world entity, even if the records do not match completely. If each record in a database or a file carried a unique, universal and error-free identification code, the only problem would be to find an optimal search sequence that would minimize the total number of record comparisons. In most cases, which are encountered in practice, the identification code of a record is neither unique nor error-free. In some of these cases, the evidence presented by the identification codes, (i.e., primary key, object id, etc.) may possibly point out that the records correspond, or that they do not correspond, to the same entity. However, in the large majority of practical problems, the evidence may not clearly point out to one or the other of these two decisions. Thus it becomes necessary to *make a decision* as to whether or not a given pair of records

must be treated as though it corresponds to the same real world entity.

The problem that we are dealing with in this paper, is also very similar to the *entity join* problem [22]. The traditional join [27], or other similar operators like theta-join, equi-join, and outer-join of the relational model, connects two tuples, if the specified fields in the operator contain the same symbol or a symbol defined by a simple comparison operator. Because of this property, this type of join is known as *symbolic join*. The *non equi-join* or *band-join* [17, 10] is a generalization of the equi-join operation, in which the join predicate requires values in the join attribute to fall within a specified band. The entity join is like the symbolic join, as it connects two tuples by matching the values on the specified columns. The difference between the symbolic and entity join lies in the concept of matching: the entity join detects a match if and only if the specified columns in the two operand rows identify the same entity. The *duplicate detection problem*, where a full pair-wise comparison of two records is performed, can also be considered as a special case of the entity join problem, and consequently, of the record matching problem.

The large volume of applications spanning the range of cases from (a) an epidemiologist, who wishes to evaluate the effect of a new cancer treatment by matching information from a collection of medical case studies against a death registry in order to obtain information about the cause and the date of death, and (b) an economist, who wishes to evaluate energy policy decisions by matching a database containing fuel and commodity information for a set of companies against a database containing the values and the types of goods produced by the companies, to (c) a computer scientist who needs to develop an algorithm for identifying mirrored hosts on the World Wide Web in order to improve web-based information retrieval, signifies the tremendous impact and applicability of the problem addressed in this paper.

The remaining part of this paper is organized as follows: Section 2 provides an overview of the record matching problem, and it explains in detail the various phases used in its solution. Section 3 provides an extensive analysis and describes metrics both quantitatively and qualitatively that can be used to evaluate the effectiveness of the record matching process in general and its phases in particular. Section 4 contains a comprehensive discussion and analysis of the techniques which have been proposed for the searching phase. Section 5 presents a number of comparison techniques which are used for computing the distance between various types of fields. In section 6, we provide the required notation used in later sections. Section 7 focuses on record matching techniques which are based on probabilistic matching. Section 8 presents various other techniques, algorithmic, distance-based and knowledge-based ones, that are widely used in the record matching phase. Section 9 provides a presentation of different commercial off the shelf tools used in industry for scrubbing data - evaluating the quality of the data, matching records and merging these records. Finally, in section 10 we discuss future directions and conclusions of the record matching area.

2 The Record Matching Problem

Record matching or linking is the process of identifying records that refer to the same real world entity or object. In the product space of two tables, a *match* M is a pair that represents the same entity and a *non-match* U is a pair that represents two different entities. Within a single table, a *duplicate* is a record that represents the same entity as another record in

the same database. Common record identifiers such as names, addresses, and code numbers (SSN, object identifier), are the *matching variables* that are used to identify matches. The vector that keeps the values of all the matching variable comparisons for a pair of records (comparison pair) is called a *comparison vector.* We denote a comparison vector by $\underline{x}$. The set of all possible vectors, is called the *comparison space* X. The final goal in the record matching is to determine whether a comparison record corresponds to a matched or a non-matched pair of database records. The presence of errors in the identifying information makes this problem hard.

The *searching* and *matching phase* are the principal phases in the record matching/linking process. The former involves searching for potentially linkable pairs of records and the latter decides whether or not a given record pair is correctly matched. A pre-processing phase is almost always required before any successful attempt is made for correctly and efficiently matching records. We call this phase the *data preparation phase* and we analyze it further later on. For the searching phase, the goal must be to reduce the number of failures to bring potentially linkable records together for comparison. The applied technique should address this problem without resorting to excessive amounts of additional searching. For the matching phase, the problem is that of devising a computer algorithm to simulate the inference needed for deciding whether or not a pair of records relates to the same entity, when some of the identifying information agrees and some disagrees.

2.1 Data Preparation

The data preparation phase includes a *parsing*, a *data transformation step*, and a *standardization step* all of which are performed first for improving the quality of the in-flow data and second for making the data comparable and more usable.

Parsing is the first critical component in the data preparation stage of record matching. This process locates, identifies and isolates individual data elements in the source files. Parsing makes it easier to correct, standardize, and match data because it allows to compare individual components, rather than long strings of data. For example, the appropriate parsing of name and address components is a crucial part in the record matching/linking process.

Data transformation refers to simple conversions that can be applied to the data in order for them to conform to the data types of their corresponding domains. As such, this category includes manipulations of data that is focused solely on one field at a time, without taking into account the values in related fields. The most common form of a simple transformation is the conversion of a data element from one data type to another. A simple data type conversion, like this, is usually required when a legacy application stored data in a data type that makes sense within the context of the application, but not in a newly developed system. Renaming of a field from one name to another is considered as a data transformation as well. Encoded values in operational systems and external data is also another problem that should be handled in this stage. This kind of values should be converted to their decoded equivalent, so as records from different sources can be compared in a uniform manner. Range checking is also another kind of data transformation which involves examining data in a field to ensure that it falls within the expected range, usually a numeric or date range. Lastly, dependency checking is slightly more involved since it requires comparing the value in a field to the values

in another field.

Data standardization refers to the process of standardizing the information represented in certain fields with some special content. This is used for information that can be stored in many different ways in different data sources and must be converted to a uniform representation before the record matching process starts. One of the most common formatting needs is for address information. There is no one standardized way to capture addresses and the same address can be represented in many different ways. Typically, when operational applications are designed and constructed, there is very little uniform handling of date and time formats across applications. Because most operational environments have many different types of dates and times, almost in every data preparation phase, we will have to transform dates and times into a standard format. Name standardization identifies components such as first names, last names, title and middle initials. Address standardization locates components such as house numbers, street names, PO Boxes, apartment numbers and rural routes. Without standardization, many true matches could erroneously be designated as non-links, based on the fact that common identifying information can not be compared.

A comprehensive and recent collection of papers related to various data transformation approaches can be found in [13].

2.2 Searching Phase

In the searching phase, there may be errors resulting from failures to bring potentially linkable pairs of records together for comparison. These errors can be reduced to zero simply by comparing each record with all the other records. This technique can be considered as *exhaustive searching* in the cross product of the data sets. Such a brute-force approach for linking records is of quadratic order, with respect to the number of records in the database. For very large databases with millions of stored records, a pair-wise comparison of all those records is not feasible. For this reason, it is common to cluster or group the database records based on the values of some grouping variables or some a-priori knowledge. This technique is called *clustering* or simply *grouping*. Clustering techniques can be very promising when there is a-priori knowledge that some of the fields in the database are error-free. This assumption is very stringent but under certain circumstances, and for a finely selected set of attributes, can be considered as feasible. 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 "single", "married" or "divorced" are clean, then the database records can be easily clustered into six different groups, e.g., the first cluster will contain all the records for unmarried males, the second cluster will contain all the married males, and so on. If we know that the values corresponding to those fields are correct, there is no possibility of any actual matches being escaped. This technique will drop the complexity of the record matching proportionately to the number of identified clusters.

Another technique which has been used for reducing the complexity of the searching process is *sorting*. The use of a sorting method with either the entire record or part of the record as the comparison key, brings similar records close to each other. The price that must be paid for the saving in complexity is an increase in the failures to bring potentially linkable pairs of records together for comparison. This is happening because it is possible that there is an error in the comparison key that is used for sorting. After the records in

the database have been sorted, a follow-up step should be applied for comparing pairs of records. This step can either be as simple as a *sequential scanning* that passes through the data only once comparing pairs of contiguous records (i.e., this is the process usually followed for duplicate detection under exact matching), or more elaborate. In the latter case, two different approaches can be applied. The first approach scans the database by comparing only those records that agree on a user-defined key, which for example can be the comparison key. The comparison key, like the key in a database table, can be used to uniquely identify a record in a database, or at least it can be used for clustering the records into groups of similar records with respect to a certain characteristic. This technique is known as *blocking*. The second approach is called *windowing* and it uses a fixed size window when checking for matches. In a windowing technique, records that fall inside a fixed size window that scans the entire database, are compared in a pair-wise fashion.

As we mentioned previously, the major disadvantage of sorting is that sorting is not complete. Basically, it leaves many actual matches undetected. A technique that partially fixes this problem is called the *multi-pass approach*. This technique performs multiple passes through the database, for sorting the records based on different sort or comparison keys each time. The results produced by using a different sort key are combined thereafter by a transitive closure stage which identifies matches in a transitive mode. For example, if record a is linked with record b in the first pass, and record b is linked with record c in a later pass, then a and c should also be compared to each other.

2.3 Matching Phase

In the matching phase, we identify two categories. The first one is called *field-based matching* and the second one *record-based matching*. Field-based matching refers to the matching of particular fields in the schema of a record. The techniques applied for field-based matching are closely related to the domain of the underlying field. The majority of the techniques in this category has been devised for strings of characters. Other techniques in this category include algorithms that test the difference in numerical (i.e., continuous and discrete) or categorical domains. Algorithms of heuristics for field-based character string matching include the calculation of the string edit distance, or various n-gram-based, string matching algorithms. For example, if we are linking person records, a possible measurement would be to compare the family names of the two records and assign the value of "1" to those pairs where there is agreement and "0" to those pairs where there is disagreement. Values in between 0 and 1 are most frequently used to indicate how close the values in the two different domains are. These measurements will yield a vector of observations on each record pair.

When pairs of records are brought together for comparison, decisions must be made as to whether these pairs are to be regarded as linked, not linked or possibly linked, depending upon the various agreements or disagreements of items of identifying information. The specification of a record linking procedure requires both a method for measuring closeness of agreement between records and a method using this measure for deciding when to classify records as matches. The former refers to the problem of assigning "weights" to individual fields of information and obtaining a "composite weight" that summarizes the closeness of agreement between two records. A scalar weight is then assigned to each record pair, thereby ordering the pairs. The latter method can be viewed as a decision process that assigns a

label that corresponds to three different possible actions. The three possible actions for each candidate pair include: to declare the two records as linked, to declare the two records as not linked, or to declare the two records as possibly-linked because of insufficient evidence. We should mention here that the possibly-linked decision is not a definite one and either further information should be acquired or the comparison pair must be manually inspected.

3 Estimating the Quality of the Record Linking Process

The requirements for time and accuracy vary among different users of record matching. According to some, the immediate emphasis should be on simplicity rather than accuracy, in the hope of getting on with the analysis sooner. Other users believe that it may be sufficient that the false positives are balanced by an equal number of failures to achieve potential good links, if such a balance can be arranged. Only for a third group, a truly high level of accuracy is important and it demands that all the potentially possible good linkages are implemented with a minimum of error or cost or time of any kind.

The record linking process is a highly resource consuming one and for this reason the person who engages in it, should be able to quantify a-priori the merit of applying this process to the data at hand. Although the accuracy of such operations and the time required are generally regarded as important, it is unusual to judge the efficiency in numerical terms. Because of that, setting down the conditions under which an optimal balance may be achieved between the level of accuracy and its cost as indicated by the time required to achieve that level, is not easy. In the following paragraphs, we discuss some important issues and numerical metrics which can be used to estimate the accuracy and time requirements for the two essential steps in the linking of records, the searching and the matching step. An experimental evaluation of various searching and matching techniques can be found in [39].

3.1 Metrics for the Searching Phase

First, we will present metrics used to estimate the effectiveness of the searching process. All of these metrics are closely related to the method used for partitioning the data source in blocks or pockets of records. In order to be able to formally define these metrics, we need to define first the presence value and the specific probability of a value. The probability that a particular field will have data in it, is the *presence* value [9]. A field is not helpful in the linking process if there is no data value in it. Since we are concerned with the presence as we compare two records to each other, the probability that they will both have data in them that can be compared is the square of the field presence of a single record. The relative frequency of occurrence for each specific value in each field is also another important measure. The relative frequency of each specific value serves as an estimate of the probability that the specific value in question occurs in the field. The frequency of each specific value is measured relative to the frequency of any value at all. This relative frequency of occurrence of a specific value in each field is called the *b-ratio* or the *specific probability*. This relative frequency can be used to estimate the probability of two records having the same value. We simply square the probability of one record to have this value. The *coincidence* value

for a field or else the probability that two records agree on any specific value in the field, is the sum of the squares of the specific probabilities. This value is called the *sum of the B squares.* Newcombe [32] used to call this measure as the *coefficient of specificity.* This is the probability that two records agree in one field, provided only that there is data in the field. The coefficient of specificity measures the fineness with which a data source will be partitioned into segments based on a particular kind of identifying information. The coefficient of specificity may be thought of as the fraction of the records falling within a cluster/segment of strictly representative size. Since most identifying information divides a set of records unevenly into a mixture of small and large segments, it is convenient to be able to indicate the effective degree of division of the data source in this simple manner.

Unlike the coefficient of specificity, which gets smaller as a data source becomes more finely divided, the *discriminating power* increases with the extent of the subdivision. Furthermore, it is usually regarded as an "addable" quantity. Thus the discriminating power may be taken as the logarithm of the inverse of the coefficient of specificity.

As we already know, every record comparison yields agreement, disagreement, or a missing value for each field. For the following measure, we are only interested in the cases where there is agreement or disagreement. The *reliability* is the relative frequency of field agreement among matched pairs [9]. The reliability is a measure of how well a field agreement helps to determine whether the records match. The record linkage community refers to this measure as the *a-ratio.* Newcombe [32] used to refer to the complement of the reliability as the *likelihood of inconsistency.* For example, suppose that we have a sample of records and that 0.5% of all the record comparisons we can possibly make are matched M. Suppose further that we choose a particular field as the predictive variable and that in observing the values in the matched record pairs, we find that some of them, say 2% of the pairs actually disagree in that field. We say that the reliability of the field is 98%.

Finally, the merit of any particular kind of identifying information for partitioning the data sources into sets of records that are prospective matches, may be taken as the ratio of the discriminating power to the likelihood of discrepancy or inconsistency of such information in linkable pairs of records. This metric is known as *merit ratio.* The most efficient partitioning of a data source will be based on the items of identifying information that have the highest merit ratios, using enough different items to achieve a combined discriminating power that will subdivide the record set to the required degree of fineness. In this way, the minimum total likelihood of discrepancy or inconsistency will have been introduced into the partitioning variables for any required degree of subdivision.

The approach permits the searching step of a linkage operation to be optimized, in terms of the numbers of (a) wasted comparisons to which an incoming record must be subjected in order to be brought together with a potentially linkable counterpart from another data source, and (b) failures to bring such records together. A tolerable level may be set of either the wasted comparisons or the failures, and the other value may then be minimized. Adjustment is achieved by adding or deleting an item from the sequencing information. In this way the fineness of subdivision is increased or decreased along with the errors simultaneously until the required balance is struck. At no time should the sequencing information include an item with a lower merit ratio where one with a higher ratio is available. The cost of the searching step is thus balanced against its precision with a view to getting the best possible bargain.

3.2 Metrics for the Matching Phase

Whether the matching status of a record comparison pair is a link, a not-link, or a possible link, depends upon the various agreements and disagreements of items of identifying information. It is also desirable that such decisions are based on numerical estimates of the degrees of assurance that the records do or do not relate to the same real world entity.

During the record matching process, we would intuitively attach greater positive weight to some of the agreements than to others and greater negative weight to some of the disagreements than to others. In each instance, the question is "Would such an agreement be likely to have occurred by chance if the pair of records did not relate to the same person?" or "Would such a disagreement be likely to have occurred by chance if the pair of records did in fact relate to the same person?". The answer in each case will depend upon prior knowledge gained by experience. For example, a middle initial known to be rare (low value specific frequency) will be regarded as less likely to agree by chance on a pair of records than a commonly occurring initial would. Similarly, when a highly reliable and stable item of identification fails to agree, it will argue more strongly that the records referred to are not the same than would.

The mathematical basis of such intuitive assessments is really quite simple. In general agreements of initials, birth dates, and such will be more common in linked pairs or records than in pairs brought together for comparison and rejected as unlinkable. The greater the ratio of these two frequencies, the greater the weight attached to the particular kind of agreement will be. In order to obtain numerical weights, that can be added to other such weights, the above ratio may simply be converted to a logarithm. In practice, the logarithm to the base two has proved particularly convenient. These weights are simply expressed as $w = \log(A, B)$ where A and B are the frequencies of the particular agreement among linked pairs of records and among pairs that are rejected as unlinkable, defined as specifically as one wishes.

The weights for agreement will have positive values because A in such circumstances is always greater than B, and these weights may be regarded as strictly analogous to the discriminating powers discussed in section 3.1. There is no need to alter the formula for the weights when deriving the weights for disagreements. A and B may be simply regarded as the frequencies of the particular disagreement, defined in any way, among linked and unlinked pairs of records. Usually the weights will then be negative in sign, because disagreements are, in most instances, less common among the linked than among the unlinked pairs.

4 Searching for Similar Records

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 in the database with all the others. 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 unlinkable. In this section we provide an extensive review of the methods which have been proposed for reducing the cost of the searching phase.

4.1 Nested-Loop Join

This method is the simplest method which can be used for joining records together and can be also applied in the record matching context. It actually follows from the definition of the join operation. One of the tables being joined is designated as the inner table and the other is designated as the outer table. For each record of the outer table all records of the inner table are read and compared with the record from the outer table. This type of join corresponds to the full pair-wise comparison of all records.

4.2 Merge-Sort, Hash Joins and Duplicate Record Elimination

The "traditional" method of eliminating duplicate records in a file has been the following: first, the file is sorted using an external merge-sort in order to bring all duplicate records together; then a sequential pass is made through the file comparing adjacent records and eliminating all but one of the duplicates. Since most operating or database systems already provide a sort facility, this approach is clearly the simplest. However, because of the expense of sorting, relational DBMSs do not always eliminate duplicates when executing a projection. Rather, the duplicates are kept and carried along in subsequent operations, thus increasing the cost of those subsequent operations. It is only after the final operation in a query is performed that the resultant relation is sorted and duplicates are eliminated.

Using any sorting method with the entire key taken as the comparison key brings identical records together. Since many fast sorting algorithms are known, sorting appears to be a reasonable method for eliminating duplicate records. The first method is an external two-way merge-sort followed by a scan that removes the duplicates. The second method for eliminating duplicate records utilizes a hash function and a bit array to determine whether two records are identical.

The success of the sort-merge join lies in the fact that it reduces the number of comparisons between records. A record from the first table is not compared with those records in the second table with which it cannot possibly join. Hash join methods achieve the same effect in another way. They attempt to isolate the records from the first table that may join with a given record from the second table. Then, records from the second table are compared with a limited set of records from the first relation.

Authors in [2] propose and evaluate an alternative approach for eliminating duplicate records from a file. In their approach a modified external merge-sort procedure is utilized and duplicates are eliminated as part of the sorting process. They also demonstrate that this approach has the potential of significantly reducing the execution time for eliminating duplicate records in a file.

4.3 Band Joins

The band join operation [17, 10] is a generalization of the equi-join operation. A non equi-join is a band join if the join predicate requires values in the join attribute of the left hand side relation to fall within a specified band about the values in the join attribute of the right hand side relation. Band joins arise in queries that require joins over continuous "real world" domains such as time or distance. For example, if the tuples in two relations represent events,

and one of the attributes from each relation represent the times at which events occur, then finding all pairs of events that occurred at nearly the same time will entail a band join.

4.4 Blocking

This is the most frequently used searching technique for matching records and it is very similar to the sorting approach. The only difference between these two techniques comes from the non-uniqueness of the sorting key for the blocking approach, in which case a sorting key is determined in an ad-hoc manner. *Blocking* is a procedure for subdividing files into a set of mutually exclusive subsets under the assumption that no matches occur across blocks. For this reason, it is usual to arrange the file in some orderly sequence by using identifying information that is common to all linkable sources. Files are ordered or blocked in particular ways in order to increase the efficiency of searching. Then, detailed comparisons should be carried out only within the small portions of the files for which the sequencing information is the same. For many purposes, it is common practice to use the alphabetic last names and first given names for sequencing a file of personal records. The price that must be paid for the saving of time is an increase in the failure to bring potentially linkable pairs of records together for comparison, owing to discrepancies in the sequencing information on pairs that in fact relate to the same person. It is important to generate blocks of the right size. The balance between the number and size of blocks is particularly important especially when large data sources are being matched. However, different kinds of information that might be used for the sequencing differ widely, both in reliability and in the extend to which they subdivide the file.

Although alphabetic last names are commonly employed, they are not particularly efficient for sequencing because of the high frequency with which they are misspelled or altered. Considerable improvement can be achieved by setting aside, temporarily, the more fallible or liable parts of the information which come from the last names, while retaining as much as possible of the inherent discriminating power. There are a number of systems for doing this. We describe below three of these systems. More detailed information about the first two can be found in [33].

4.4.1 Soundex Code

The most common coding scheme 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. The rules of Soundex coding are as follows: (i) The first letter of a surname is not coded and serves as the prefix letter. (ii) W and H are ignored completely. (iii) A, E, I, O, U and Y are not coded but serve as separators (see v below). (iv) Other letters are coded as follows, until three digits have been used up (the remaining letters are ignored): B, F, P, V, coded 1; D, T coded 3; L coded 4; M, N coded 5; R coded 6; all other consonants are (C, G, J, K, Q, S, X, Z) coded 2. (v) Exceptions are letters which follow letters having the same code, or prefix letters which would, if coded, have the same code. These are ignored in all cases unless a separator (see iii above) precedes them. Newcombe [32] reports that the Soundex code remains unchanged with about two-thirds of the spelling variations observed in linked pairs of vital records, and that it sets aside only

a small part of the total discriminating power of the full alphabetic surname. The code is designed primarily for Caucasian surnames, but works well for files containing names of many different origins (such as those appearing on the records of the U.S. Immigration and Naturalization Service). However, this code is less satisfactory, where the files contain names of predominantly Oriental origin, because much of the discriminating power of these names resides in the vowel sounds, which the code ignores.

4.4.2 NYSIIS

The New York State Information and Intelligence System (NYSIIS) differs from Soundex in that it retains information about the position of vowels in the encoded word by converting all vowels to the letter A. It also returns a purely alpha code (no numeric component). The NYSIIS coding proceeds in well defined steps. The first letter(s) of a name are tested and changed as necessary. The same is done for the last letter(s) of the name. Then, starting with the second letter, a scan is carried out of each letter of the name using an imaginary pointer and, at each step, changes may be made to the name according to a set of rules called a "loop". The rules are applied each time the pointer is moved to the next letter of the name. During the process, characters may be taken from the altered name and transferred to create the NYSIIS code. Finally the end portion of the NYSIIS code as formed in this manner, is subjected to a further test and is changed as necessary. Usually the NYSIIS code for a surname is based on a maximum of nine letters of the full alphabetical name, and the NYSIIS code itself is then limited to six characters.

4.4.3 ONCA

A recent development, referred to as Oxford Name Compression Algorithm (ONCA), uses an anglicized version of the NYSIIS method of compression as the initial or pre-processing stage, and the transformed and partially compressed name is then Soundexed in the usual way. This two-stage technique has been used successfully for blocking files, and overcomes most of the unsatisfactory features of pure Soundexing while retaining a convenient four-character fixed-length format.

4.5 Entity Grouping

The following approach has been used specifically for identifying author-title clusters in bibliographic records [20]. The algorithm for identifying clusters of related records considers each record in the collection and determines whether it has (approximately) the same author and title field as any other record. The algorithm uses two rounds of comparisons to identify author-title clusters. The first round creates a pool of potentially matching records by using a full-text search of the entire collection; the pool consists of the results of three queries, with values randomly selected from the author and title of the source record. Three different queries are used to construct the potential match pool. Each query includes one of the authors's last names and two words from the title list. Title words that are either one to two letters long, or in the stop-list, are not used. In cases where there are not enough words to construct three full queries, queries use fewer words.

In the second round, each potential match is compared to the source record and an author-title cluster is created for the matching records. In this phase, an n-gram based approximate string matching algorithm is used to compare the two fields. The algorithm is applied to every source record, regardless of whether it has been placed in a cluster already; thus a cluster with five records will be checked by five different passes. The algorithm also enforces transitivity between records: if record r_1 matched record record r_2, and record r_2 matched record r_3, all three would be placed in the same cluster, whether or not r_1 matches r_3.

4.6 Sorted Neighborhood Approach

The authors in [19] describe the so-called sorted neighborhood approach. Given a collection of two or more databases, the databases are first concatenated into a sequential list of records, after the records have been standardized. The sorted-neighborhood method, for searching of linkable comparison pairs, is summarized in the following three steps:

Create keys A key for each record in the list is computed by extracting relevant fields or portions of fields.

Sort data The records in the database are sorted by using the key found in the first step.

Merge A fixed size window is moved through the sequential list of records in order to limit the comparisons for matching records to those records in the window. If the size of the window is w records then every new record that enters that window is compared with the previous $w-1$ records to find "matching" records. The first record in the window slides out of it.

Although sorting the data may not be the dominant cost of the sorted-neighborhood approach, the authors consider an alternative approach for sorting the entire database. They propose to partition the dataset into independent clusters first by using a key that is also extracted from the data. For each one of the clusters which are generated in that way, the sorted-neighborhood method can be applied. Clustering data (very similar to blocking presented above) raises the issue of how well the data are partitioned after being clustered.

4.7 Priority Queue Algorithm

Under the assumption of transitivity, the problem of matching records in a database can be described in terms of determining the connected components of an undirected graph [29]. At any time, the connected components of the graph correspond to the transitive closure of the "record matches" relationships discovered so far. There is a well-known data structure that efficiently solves the problem of determining and maintaining the connected components of an undirected graph, called the union-find data structure. This data structure keeps a collection of disjoint updatable sets, where each set is identified by a representative member of the set. The data structure has two operations:

- `Union(x,y)` combines the sets that contain node x and node y say S_x and S_y, into a new set that is their union $S_x \cup S_y$. A representative for the union is chosen and the new set replaces S_x and S_y in the collection of disjoint sets.

- `Find(x)` returns the representative of the unique set containing x. If `Find(x)` is invoked twice without modifying the set between the requests, the answer is the same.

The proposed algorithm uses two passes. The first one treats each record as one long string and sorts all the records lexicographically, by reading from left to right whereas the second pass sorts the records by reading them from right to left. The algorithm uses a priority queue of sets of records that belong to the last few clusters detected. The algorithm scans the database sequentially and determines whether each record scanned is or is not a member of a cluster already represented in the priority queue. To determine cluster membership, the algorithm uses the `Find` operation that was previously described. If the record is already a member of a cluster in the priority queue, then the next record is scanned. If the record is not already a member of any cluster kept in the priority queue, then the record is compared to representative records in the priority queue by using a domain independent algorithm. If one of these comparisons succeeds, then the record belongs in this cluster and the `Union` operation is performed on the two sets. On the other hand, if all comparisons fail, then the record must be a member of a new cluster not currently represented in the priority queue. Thus, the record is saved in the priority queue as a singleton set.

The set that represents the cluster with the most recently detected cluster member has highest priority in the queue. The priority queue contains a fixed number of sets of records. Each set contains one or more records from a detected cluster. Intuitively, the sets in the priority queue represent the last few cluster detected. For this reason, if the insertion of a new singleton set causes the size of the priority queue to exceed its limit the lowest priority set is removed.

4.8 Multi-Pass Approach

The effectiveness of the sorted neighborhood approach, that was described above, depends highly on the comparison key that is selected to sort the records. A sorting key is defined to be a sequence of attributes, or a sequence of sub-strings within the attributes, chosen from the record in an ad hoc manner. In general, no single key will be sufficient to sort the records in such a way that all the matching records can be detected. Attributes that appear first in the key have a higher priority than those that appear following them. If the error in a record occurs in the particular field or portion of the field that is the most important part of the sorting key, there is a very small possibility that the record will end up close to a matching record after sorting.

To increase the number of similar records merged, the authors in [19] implemented a strategy for executing several independent runs of the sorted-neighborhood method (presented above) by using each time a different sorting key and a relatively small window. This strategy is called the *multi-pass* approach. Each independent run produces a set of pairs of records that can be merged. The transitive closure of the records matched in different passes is subsequently computed. The results will be the union of all different matching pairs

discovered in all independent runs plus all those pairs that can be inferred by transitivity of equality.

4.9 Incremental Merge-Purge

The authors in [19] propose an incremental technique and claim to be efficient in consecutive updates of the same integrated data source. Their algorithm specifies a loop repeated for each increment of information received by the system. The increment is concatenated to a table of *prime representatives* that is computed during the previous (last) run of the incremental merge-purge algorithm. One prime representative is selected from each cluster, identified in one pass, in such a way that it represents the information of this cluster in future updates. Of particular importance to the success of this incremental procedure, in terms of accuracy of the results, is the effective selection of the prime representatives.

4.10 Parallel Merge-Purge

The authors in [18] propose an implementation of the original sorted neighborhood approach using a parallel shared-nothing multiprocessor computer system. For the sorting step, the input data is fragmented among the available processors. Each processor sorts its local part in parallel and then a p-way join is performed (P is the number of processors). Each processor merges its local fragment according to the specified window size separately. In order for each processor to work independently (in parallel) each segment includes the last $w-1$ records and the first $w-1$ from its previous and its next segment correspondingly. For a very large database and a reasonable number of available processors, the replication of the records in the processors is not a serious problem.

5 Field Matching and String Matching Techniques

Since it is commonplace for different strings to exhibit typographical variation, the record linking process needs effective string comparison techniques that deal with typographical variations. Dealing with typographical errors can be very important in a record linkage context. If the comparisons of pairs of strings are done only in an exact character-by-character manner, many matches may be lost.

A list of different techniques, that have been applied for field matching in the record linkage context, is shown below:

- Edit distance: We assume that the characters come from a finite set, English characters for example, and that A and B are two strings of characters. We would like to change A, character by character, such that it becomes equal to B. We allow three types of changes, or edit steps, and we assign a cost of one to each: (1) insert - insert a character into the string, (2) delete - delete a character from the string, and (3) replace - replace one character with a different character. For example, in order to change the string "abbc" into the string "babb", we can delete the first a, then insert an a between the two b's, and then replace the last c with a b for a total of three changes. However, we

can also insert a new b at the beginning and then delete the last c, for a total of two changes. Our goal is to minimize the number of single-character changes. Information about algorithms (i.e., dynamic programming) that solve this problem can be found in various algorithm textbooks [26].

- A basic field matching algorithm: The authors in [28] propose a basic algorithm for matching text fields. Based on their algorithm, a simple definition of the degree to which two fields match is the number of their matching atomic strings divided by their average number of atomic strings. An atomic string is a sequence of alphanumeric characters delimited by punctuation characters. Two atomic strings match if they are the same string or if one is the prefix of the other. The algorithm to compute the basic matching score is straightforward. First, the atomic strings of each field are extracted and sorted and second, each atomic string of one field is searched in the other fields list of strings. The number of matched atomic strings is recorded.

- Smith-Waterman: Given two strings, the Smith-Waterman algorithm uses dynamic programming to find the lowest cost series of changes that converts one string to another. This is the minimum edit distance weighted by the cost between the strings. Much of the power of the Smith-Waterman algorithm is due to its ability to introduce gaps in the records. A gap is a sequence of non-matching symbols. The Smith-Waterman algorithm has three parameters. The first one is a matrix of match scores for each pair of symbols in the alphabet. From the remaining two parameters that affect the start and the length of a gap, one is related to the cost of starting a gap in an alignment, and the other to the cost of continuing the gap. The Smith-Waterman algorithm computes a score matrix. An entry in this matrix is the best possible matching score between a prefix from one string and another prefix from the other string. When the prefixes match exactly, the optimal alignment can be found along the main diagonal. For approximate matches, the optimal alignment can be found within a small distance off the diagonal.

- N-grams: Letter n-grams, including trigrams, bigrams, and/or unigrams, have been used in a variety of ways in text recognition and spelling correction techniques [25]. Following the definition in [20] an n-gram is a vector representation that includes all the n-letter combination in a string. The n-gram vector has a component vector for every possible n-letter combination. The n-gram representation of a string has a non-zero component vector for each n-letter substring it contains, where the magnitude is equal to the number of times the substring occurs. The string comparison algorithm forms n-gram vectors (the author in [20] uses trigrams) for the two input strings and subtracts one vector from the other. The magnitude of the resulting vector difference is compared to a threshold value. If the magnitude of the difference is less than the threshold, the two strings are declared to be the same. The threshold value is determined experimentally by using several dozen pairs of strings that have been manually labeled as same or different, based on how the algorithm should compare them.

- Jaro algorithm: Jaro [21] introduced a string comparison algorithm that accounts for insertions, deletions, and transpositions. The basic Jaro algorithm includes the follow-

ing steps: (1) compute the string lengths, (2) find the number of common characters in the two strings, and (3) find the number of transpositions. The definition of common denotes the agreeing characters that are within half of the length of the shorter string. A transposition refers to those characters from one string that are out of order with respect to their corresponding common characters from the other string. The string comparison value is $\text{jaro}(s1, s2) = \frac{1}{3}(\frac{\text{common}}{\text{strlen}_1} + \frac{\text{common}}{\text{strlen}_2} + 0.5\frac{\text{transpositions}}{\text{common}})$ where s_1, s_2 are the strings with lengths strlen_1, strlen_2, respectively.

- Text similarity measure: The authors in [35] propose the so-called text similarity measure. For a given text string, they use a vector to represent all the characters of the string with the consecutive space characters collapsed into one. For two given character sequences, their objective is to find a mapping in such a way that a character similarity function that they define is maximized. The optimization problem generated in this way, can be solved efficiently by a dynamic time warping method.

- Last name coding (Soundex, etc.): Phonetic coding schemes, like Soundex [32, 33] (Section 4.4.1), are employed most frequently to bring together variant spellings of what are essentially the same names. Occasionally, phonetic coding is used instead to reveal similarities between names even where the codes themselves may differ.

- Recursive field matching algorithm: The authors in [28] propose a recursive field matching algorithm, that takes into account the recursive structure of typical textual fields. For a pair of strings, the base case is that the strings match with degree 1.0, if they are the same atomic string or one abbreviates the other; otherwise their degree of match is 0.0. Each subfield of a string is assumed to correspond to the subfield of the other string with which it has the highest score. The score of matching two strings then equals the mean of these maximum scores.

6 Notation

Given n fields, and a sample of N record pairs drawn from the Cartesian product of two tables (A and B) $A \times B$, let $x_i^j = 1$ if field i agrees for record pair j, and let $x_i^j = 0$ if field i disagrees for record pair j, for $i = 1, \ldots, n$ and $j = 1, \ldots, N$. The input to the record matching decision process is a random vector with n components of ones and zeros showing field agreements and disagreements for the j-th pair in the sample $\underline{x}^j = [x_1, x_2, \ldots, x_n]^T$, where T denotes the transpose of the vector. When it is clear from the context, we drop the superscript from the random vector $\underline{x}$. A random vector may be characterized by a distribution probability function, which is defined by $P(x_1, x_2, \ldots, x_n) = Pr\{x_1 \leq x_{10}, x_2 \leq x_{20}, \ldots, x_n \leq x_{n0}\}$, where $Pr(A)$ is the probability of an event A. Another expression for characterizing a random vector is the density function $p(\underline{x}) = \partial^n P(\underline{x})/\partial x_1 \ldots \partial x_n$. In the record matching problem, we deal with random vectors drawn from two classes or categories, each of which is characterized by its own density function. This density function is called the class i density or conditional density of class i, and is expressed as $p(\underline{x}|C)$ where C can take on the values of M or U. The a-priori probability of a class is expressed as π_C. Since there are only two classes in the record matching problem, the following equality holds $\pi_M + \pi_U = 1$. The unconditional

density function of a comparison vector $\underline{x}$, which is sometimes called the mixture density function, is given by $p(\underline{x}) = \pi_M \cdot p(\underline{x}|M) + \pi_U \cdot p(\underline{x}|U)$. The a posteriori probability of a class C, given the comparison vector $\underline{x}$ which is expressed as $p(C|\underline{x})$, can be computed by using the Bayes theorem as follows: $p(C|\underline{x}) = \pi_C \cdot p(\underline{x}|C)/p(\underline{x})$.

7 Probabilistic Record Matching Models

Newcombe was the first to recognize the record matching as a statistical problem [31]. Through an observation or measurement process, we obtain a set of numbers which make up the comparison vector. The comparison vector serves as the input to a decision rule by which the sample is assigned to one of the given classes. We first assume that the comparison vector is a random vector whose conditional density function depends on its class. If the conditional density function for each class is known, the record matching problem becomes a problem in statistical hypothesis testing.

7.1 Hypothesis Tests

In order to introduce the solution methodology in the record matching problem, we start with two-class/two-region problems. These problems arise because of the fact that the sample belongs to one of two classes and the problem is to decide how to assign the sample to either one of these classes. The conditional density functions and the a-priori probabilities are assumed to be known at this point. In the sequel, we will discuss various techniques that have been developed for addressing this decision problem.

7.1.1 The Bayes Decision Rule for Minimum Error

Let $\underline{x} \in X$ be a comparison vector, randomly drawn from the comparison space, and let our purpose be to determine whether $\underline{x}$ belongs to M or U. A decision rule, based simply on probabilities, may be written as follows: if $p(M|\underline{x}) \geq p(U|\underline{x})$ then it is decided that $\underline{x}$ belongs to M, otherwise it is decided that $\underline{x}$ belongs to U. This decision rule indicates that if the probability of the match class M, given the comparison vector $\underline{x}$, is larger than the probability of the non-match class U, $\underline{x}$ is classified to M, and vice versa. By using the Bayes theorem we can calculate the a posteriori probability from the a priori probability and the conditional density function. So the previous decision rule may be expressed as $\pi_M \cdot p(\underline{x}|M)/p(\underline{x}) \geq \pi_U \cdot p(\underline{x}|U)/p(\underline{x})$. Since the mixture density function $p(\underline{x})$ is positive and common to both sides of the inequality, the decision rule can be expressed as $\pi_M \cdot p(\underline{x}|M) \geq \pi_U \cdot p(\underline{x}|U)$ or as $l(\underline{x}) = \frac{p(\underline{x}|M)}{p(\underline{x}|U)}) \geq \frac{\pi_U}{\pi_M}$. The term $l(\underline{x})$ is called the *likelihood ratio*. The ratio π_U/π_M denotes the threshold value of the likelihood ratio for the decision.

The decision rule that it was described is called the *Bayes test* for *minimum error*. In general, the decision rule presented, or any other decision rule which is based on probabilities, does not lead to perfect classification. In order to evaluate the performance of a decision rule, we must calculate the probability of error, that is the probability for a sample to be assigned to the wrong class. It can be easily shown that the Bayes test, that we presented is a decision rule which gives the smallest probability of error.

7.1.2 The Bayes Decision Rule for Minimum Cost

Often in practice, the minimization of the probability of the error is not the best criterion to design a decision rule as the misclassifications of M and U samples may have different consequences. Therefore, it is appropriate to assign a cost c_{ij} to each situation, which is the cost of deciding that $\underline{x}$ belongs to the i-th class when $\underline{x}$ actually belongs to the j-th class. Let us denote the assignment of the class M and U to a random comparison vector, as decision A_1, and A_2 respectively. Then, the conditional cost of deciding that $\underline{x}$ belongs to the i-th class given $\underline{x}$ is $r_i(\underline{x}) = c_{iM} \cdot p(M|\underline{x}) + c_{iU} \cdot p(U|\underline{x})$. The decision rule for assigning a random comparison vector, to either one of the two classes, becomes then: if $r_1(\underline{x}) < r_2(\underline{x})$ then assign the random vector to class M, otherwise to class U. It can be easily proved that the minimum cost decision rule for the two class and two regions record matching problem can be stated as follows: if $l(\underline{x}) > \frac{(c_{12}-c_{22})\cdot\pi_U}{(c_{21}-c_{11})\cdot\pi_M}$ then assign the random comparison vector $\underline{x}$ to M, otherwise to U. Comparing the minimum error and minimum cost decision rule, we notice that the Bayes test for minimum cost is a likelihood ratio test with a different threshold from the Bayes test for minimum cost, and that the selection of the cost functions is equivalent to changing the a-priori probabilities π_M and π_U. The two decision rules become the same for the special selection of the cost functions $c_{21} - c_{11} = c_{12} - c_{22}$. In this case, the cost functions are called symmetrical. For a symmetrical cost function, the cost becomes the probability of error and the Bayes test for minimum cost minimizes exactly this error. Different cost functions are used when a wrong decision for one class is more critical than a wrong decision for the other class.

7.2 Hypothesis Tests with a Reject Region

When the conditional error, given the random comparison vector $\underline{x}$, is close to 0.5, the conditional error of making the decision given $\underline{x}$ is high. So, we could postpone the decision-making and call for a further test. This option is called *reject*. By setting a threshold for the conditional error, we may define the *reject region* and the *reject probability*.

In 1950's Newcombe introduced concepts of record matching that were formalized in the mathematical model of Fellegi and Sunter, as we have already mentioned. Newcombe recognized that record matching is a statistical problem. Fellegi and Sunter formalized this intuitive recognition by defining a linkage rule as a partitioning of the comparison space into the so-called "linked" subset, a second subset for which the inference is that the comparison vector refers to different underlying units and a complementary third subset where the inference cannot be made without further evidence.

In order to define the linkage rule, they started by defining a unique ordering of the finite set of possible realizations of the comparison vector $\underline{x}$. If any value of $\underline{x}$ is such that both $p(\underline{x}|M)$ and $p(\underline{x}|U)$ are equal to zero, then the unconditional probability of realizing that value of $\underline{x}$ is equal to zero, and hence it is not necessary for these samples to be included in the comparison space X. They also assigned an order arbitrarily to all $\underline{x}$'s for which $p(\underline{x}|M) > 0$ but $p(\underline{x}|U) = 0$. Next, they re-ordered all remaining $\underline{x}$'s in such a way that the corresponding sequence of the likelihood ratio $l(\underline{x})$ is monotone decreasing. When the value of the ratio was the same for more than one $\underline{x}$, they ordered those $\underline{x}$'s arbitrarily. After indexing the ordered set X by the subscript i, and by writing the conditional densities as

$m_i = p(\underline{x}|M)$ and $u_i = p(\underline{x}|U)$ they chose index values n and n' such that

$$\sum_{i=1}^{n-1} u_i < \mu \leq \sum_{i=1}^{n} u_i \tag{1}$$

$$\sum_{i=n'}^{N_X} m_i \geq \lambda > \sum_{i=n'+1}^{N_X} m_i \tag{2}$$

for a pair of error levels (μ, λ), where N_X is the number of points in X. Fellegi and Sunter showed that the decision rule is optimal in the sense that for any pair of fixed upper bounds on the rates of false matches and false non-matches, the manual/clerical review region is minimized over all decision rules on the same comparison space X.

Tepping [38]was the first to propose a solution methodology focusing on the costs of the decision. Verykios and Moustakides in [41] developed a cost model for record matching by using three decision areas. Without loss of generality, let us denote by c_{ij} the cost of taking a decision A_i when the comparison record corresponds to some pair of records with actual matching status j. Given the a-priori matching probabilities and the various misclassification costs, the mean cost can be written as follows:

$$\begin{aligned}\bar{c} = & \sum_{\underline{x} \in A_1} [f_M \cdot c_{10} \cdot \pi_0 + f_U \cdot c_{11} \cdot (1 - \pi_0)] \\ & + \sum_{\underline{x} \in A_2} [f_M \cdot c_{20} \cdot \pi_0 + f_U \cdot c_{21} \cdot (1 - \pi_0)] \\ & + \sum_{\underline{x} \in A_3} [f_M \cdot c_{30} \cdot \pi_0 + f_U \cdot c_{31} \cdot (1 - \pi_0)].\end{aligned} \tag{3}$$

Every point $\underline{x}$ in the decision space A, belongs either in partition A_1, or in A_2 or in A_3 and it contributes additively in the mean cost $\bar{c}$. We can thus assign each point independently either to A_1, or A_2 or A_3 in such a way that its contribution to the mean cost is minimum. This will lead to the optimum selection for the three sets which we denote by A_1^o, A_2^o, and A_3^o. Based on this observation, a point $\underline{x}$ is assigned to one of the three optimal areas if its cost for the selected area is less than its cost in both of the remaining areas. This leads to the selection of the following clustering of the decision space A:

$$A_1^o = \{\underline{x} : \frac{f_U}{f_M} \leq \frac{\pi_0}{1 - \pi_0} \cdot \frac{c_{30} - c_{10}}{c_{11} - c_{31}} \text{ and, } \frac{f_U}{f_M} \leq \frac{\pi_0}{1 - \pi_0} \cdot \frac{c_{20} - c_{10}}{c_{11} - c_{21}}\} \tag{4}$$

$$A_2^o = \{\underline{x} : \frac{f_U}{f_M} \geq \frac{\pi_0}{1 - \pi_0} \cdot \frac{c_{20} - c_{10}}{c_{11} - c_{21}} \text{ and, } \frac{f_U}{f_M} \leq \frac{\pi_0}{1 - \pi_0} \cdot \frac{c_{30} - c_{20}}{c_{21} - c_{31}}\} \tag{5}$$

$$A_3^o = \{\underline{x} : \frac{f_U}{f_M} \geq \frac{\pi_0}{1 - \pi_0} \cdot \frac{c_{30} - c_{10}}{c_{11} - c_{31}} \text{ and, } \frac{f_U}{f_M} \geq \frac{\pi_0}{1 - \pi_0} \cdot \frac{c_{30} - c_{20}}{c_{21} - c_{31}}\} \tag{6}$$

The authors also proved that the model is cost optimal in the sense that it minimizes the mean cost of a decision. They also computed the probability of the two types of errors. Their approach can be easily extended to a decision model with more than three decision

areas. The main problem for such a generalization is to appropriately order the thresholds which determine the regions in such a way that no region disappears. Similar studies have been presented in [38, 11].

7.3 Model Parameter Estimation by Using the EM Algorithm

Applying the solution methodology, presented previously, involves determining estimates of the conditional probabilities $m_i = p(\gamma_i = 1|M)$ and $u_i = p(\gamma_i = 1|U)$ that are the probabilities that a specific record pair agrees on the field i given that it is a *match* or *unmatch* pair respectively. Jaro [21] made use of the EM algorithm, which is described below.

EM algorithm is one of the common used algorithms for estimating these probabilities. EM algorithm considers estimating a family of parameters ϕ for a data set x given an incomplete version of this data set y. Postulating a family of sampling densities $f(x|\phi)$ and deriving its corresponding family of sampling densities $h(y|\phi)$, the EM algorithm is directed at finding a value of ϕ which maximizes $h(y|\phi)$. A detailed description of the EM algorithm in general may be found at [8].

In our model, the complete data set is the vector $< \gamma, g >$ such that γ is the whole set of comparison vectors we have already, and g is the vector $(1,0)$ or $(0,1)$ according to whether this pair is matched or not. The parameters to estimate are $\phi = (m, u, p)$ such that p is the proportion of the matched pairs $|M|/|M \cup U|$. The incomplete data we have is only γ since the whole goal of the model is to discover g. Then, the complete data log-likelihood is $\ln f(x|\phi) = \sum_{j=1}^{N} g_j \cdot (\ln \Pr\{\gamma^j|M\}, \ln \Pr\{\gamma^j|U\})^T + \sum_{j=1}^{N} g_j \cdot (\ln p, \ln(1-p))^T$ such that N is the total number of vectors γ^j each of which represents a record pair. The EM algorithm applies several iterations till the convergence of the estimated values m_i and u_i. Each iteration calculates the following:

Step 1

$$g_m(\gamma^j) = \frac{p \prod_{i=1}^{n} m_i^{\gamma_i^j} (1-m_i)^{1-\gamma_i^j}}{p \prod_{i=1}^{n} m_i^{\gamma_i^j} (1-m_i)^{1-\gamma_i^j} + (1-p) \prod_{i=1}^{n} u_i^{\gamma_i^j} (1-u_i)^{1-\gamma_i^j}}.$$

$$g_u(\gamma^j) = \frac{(1-p) \prod_{i=1}^{n} u_i^{\gamma_i^j} (1-u_i)^{1-\gamma_i^j}}{p \prod_{i=1}^{n} m_i^{\gamma_i^j} (1-m_i)^{1-\gamma_i^j} + (1-p) \prod_{i=1}^{n} u_i^{\gamma_i^j} (1-u_i)^{1-\gamma_i^j}}$$

Step 2

$$m_i = \frac{\sum_{j=1}^{N} \gamma_i^j g_m(\gamma^j)}{\sum_{j=1}^{N} g_m(\gamma^j)}, \quad u_i = \frac{\sum_{j=1}^{N} \gamma_i^j g_u(\gamma^j)}{\sum_{j=1}^{N} g_u(\gamma^j)}$$

Step 3

$$p = \frac{\sum_{j=1}^{N} g_m(\gamma^j)}{N}$$

such that γ_i^j equals 1 or 0 according to whether the field i in record pair j agrees or not, and n is the number of fields. The EM algorithm is highly stable and the least sensitive to the initial values.

7.4 A Decision Model for Presence and Agreement

A more detailed decision model was proposed by DuBois in [34]. Let p_j be the probability that the j-th corresponding item on the records a and b is present when the outcome of the comparison (a, b) is a true linkage, and let p_j^* be similarly defined when the outcome is a true non-linkage. Likewise, let q_j be the probability that the j-th corresponding item on the records a and b is identical when the outcome of the comparison (a, b) is a true linkage and let q_j^* be similarly defined when the outcome is a true non linkage. By defining the variables X_j and Y_j in the following manner:

$$X_j = \begin{cases} 1 & \text{if the } j\text{-th corresponding item on both records is present,} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

$$Y_j = \begin{cases} 1 & \text{if the } j\text{-th corresponding item on both records is identical,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

then

$$X_j Y_j = \begin{cases} 1 & \text{if the } j\text{-th corresponding item on both records} \\ & \text{is present and identical,} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

which implies that Y_j is observable when $X_j = 1$. We use $X_j Y_j$ in order to distinguish between when $y_j = 0$ and when Y_j is not observable due to $X_j = 0$. Consider the $2L$-dimensional random vector

$$X^* = (X_1, X_2, \cdots, X_L; X_1 Y_1, X_2 Y_2, \cdots, X_L Y_L) \tag{10}$$

defined so that the outcome of any comparison (a, b) is a point $X^*(a, b) = \underline{x}^*$ where:

$$\underline{x}^* = (x_1, x_2, \cdots, x_L; x_1 y_1, x_2 y_2, \cdots, x_L y_L) \tag{11}$$

Let the random vectors $X' = (x_1, x_2, \cdots, X_L)$ and $Y' = (Y_1, Y_2, \cdots, Y_L)$ have independent components and suppose the X_j's and Y_j's are distributed as point binomials $b(1, p_j)$ and $b(1, q_j)$ on M and $b(1, p_j^*)$ and $b(1, q_j^*)$ on U. If the components $X_1, X_2, \cdots, X_L$ and $X_1 Y_1, X_2 Y_2, \cdots, X_L Y_L$ of the random vector X^* are mutually independent within each sequence, then the multivariate probability mass functions of the random vector X^* under M and U is given by:

$$f_1 = f(x^*|M) = \prod_{j=1}^{L} p_j^{x_j} (1 - p_j)^{1 - x_j} q_j^{x_j y_j} (1 - q_j)^{(1 - y_j) x_j} \tag{12}$$

and

$$f_2 = f(x^*|U) = \prod_{j=1}^{L} p_j^{*x_j}(1-p_j^*)^{1-x_j} q_j^{*x_j y_j}(1-q_j^*)^{(1-y_j)x_j} \quad (13)$$

7.5 Matching and Sequential Analysis

As the record comparison process is computationally very expensive, its application should be economical. For example, it maybe the case that after having computed the differences of a small part of the fields of two records, it is obvious that the pair does match, no matter what the result of the remaining comparisons will be. So it is paramount to determine the field comparison for a pair of records as soon as possible so as to save valuable computation time. The field comparisons should be ended when even complete agreement of all the remaining fields cannot reverse the unfavorable evidence for the matching of the records [33]. To make the early cut-off work, the global likelihood ratio for the full agreement of each of the identifiers should be calculated. At any given point in the comparison sequence, the maximum collective favorable evidence, that could be accumulated from that point towards, will indicate what improvement in the overall likelihood ratio might conceivably result if the comparisons were continued. A similar study has been presented in [11] where the authors draw upon the research in the area of sequential decision making in order to avoid the communication overhead of copying data from remote places to some local place for processing. The main benefit in that case study was that not all the attributes of all the remote records are brought to the local site; instead attributes are brought one at a time. After acquiring an attribute, the matching probability is revised based on the realization of that attribute, and a decision is made whether or not to acquire more attributes.

8 Non-Probabilistic Record Matching Models

In this section, various non-probabilistic approaches, that have been developed specifically for solving the record matching problem, are described. We have collected different approaches in this category, mostly knowledge-based, distance-based, induction-based as well as learning both supervised and un-supervised.

8.1 Domain-independent Matching

The authors in [28, 29] propose a domain independent algorithm for detecting approximately duplicate database records. They call a record matching algorithm domain-independent if the algorithm can be used without any modifications in a range of applications. The basic idea is to apply a general purpose field matching algorithm , especially one that is able to account for gaps in the strings, to play the role of the record matching algorithm.

8.2 Equational Theory Model

The authors in [19] build an equational theory that dictates the logic of domain equivalence. This equational theory specifies the inference about the similarity of the records. For ex-

ample, if two persons have their names spelled nearly enough and these persons have the same address, we may infer that they are the same person. Specifying such inference in the equational theory requires a declarative rule language. For example, the following is a rule that exemplifies one axiom of the equational theory developed for an employee database:

```
FORALL (r1,r2) in EMPLOYEE
IF r1.name is similar to r2.name AND r1.address = r2.address
THEN r1 matches r2
```

Note that "similar to" is measured by one of the string comparison techniques (Section 5), and "matches" means to declare that those two records are matched and that represent the same person.

8.3 Supervised Learning

Authors in [7] describe a complete data reconciliation methodology to be used for approximate record matching. The uniqueness in their approach relies on the incorporation of machine learning and statistical techniques for reducing the matching complexity for large datasets. The reduction was achieved by pruning parameters that do not contribute much to the matching accuracy while at the same time, the matching complexity is increased.

Their approach is based on selecting a machine learning algorithm to generate matching rules. After a specific algorithm has been selected, parameters are pruned to yield a matching rule of low complexity. Once the improved matching rule has been developed on a sample dataset, it can be applied to the original large dataset.

The authors experimented with three different learning techniques. They used the well-known CART [3], algorithm that generates classification and regression trees, a linear discriminant algorithm, which generates linear combination of the parameters for separating the data according to their classes, and the so-called vector quantization approach, which is a generalization of nearest neighbor algorithms.

The comparison space that it is used in their experiments contain the usual parameters such as the names and addresses of customers along with record descriptors of the records being compared, e.g., field lengths. The main goal in the experiments was the matching of customer's records from a wire-line and a wireless telecommunication database. They concentrated on a very small sample of the data in order to build the matching rules by generating a table of prospective matches using a simple matching algorithm over the sampled dataset.

The experiments which were conducted indicate that CART has the smallest error percentage. The rates were based on a sample of 50 averaged runs over the dataset, where half of the dataset was used to build the matching rule and the other half was used to judge the performance. The selection of the CART algorithm, for generating the matching rules in the dataset, was also based on one of the features of the generated rules by the CART algorithm, which is in conjunctive normal form, so that it could be used for building separate indices or clusters for each parameter. After they used CART to generate the matching rules, they also applied a model selection method for selecting the set of parameters that could achieve a good trade-off between classification accuracy and the complexity of the matching model generated by the machine learning algorithm.

The authors reported that the rules have had a very high accuracy, and low complexity. The automatically reported rules increased the accuracy of the matching process from 63% to 90%. The pruning generated by the model selection approach adopted reduced the complexity by over 50% by eliminating a proportionally large number of parameters with respect to the initially induced model.

8.4 Distance-Based Techniques

Probability models require one to accurately estimate the probability parameters and counts from training data. When manually matched training data are readily available, these probability parameters can be estimated in a relatively straightforward manner. However, in the absence of such training data, one cannot easily estimate these probability parameters. As a result, the probability based decision model is not appropriate in those situations. The authors in [12] overcome this limitation by using a distance-based measure in order to express the similarity between two records, and then use the distance in a decision theoretic setting. The distance between two records is expressed as a weighted sum of the distances between their attribute values. The weights used in the model, parameterize the predictive power of an attribute in terms of its discriminating power. Those distances, are then used in a simple assignment model.

8.5 Unsupervised Learning

The authors in [40] have contributed in automating the matching rule generation process by incorporating machine learning techniques such as clustering and classification. As we said before, the comparison space consists of comparison vectors which contain information about the differences between fields of a pair of records. Unless some information exists about what comparison vector correspond to which category (link, non-link or possible-link), the labeling of the comparison vectors should be done manually.

In order to solve the problem of labeling the comparison vectors, the authors use a technique to identify clusters of comparison vectors based on the intuition that comparison vectors corresponding to similar record pairs will fall under the same cluster. The AutoClass [6] clustering tool was used for the partitioning of the comparison space. AutoClass uses probability models for describing the classes, and it supports two kinds of models. The one used in the study was the multi normal model that 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. The clusters so derived, are automatically mapped to the three values of link, non-link and possible link.

By combining the cluster/label values with their corresponding comparison vectors, they could create a set of training vectors to be given as input to a learning algorithm, such as ID3 [36] which induces decision trees from labeled examples.

8.6 Model and Feature Selection

For reducing the complexity of processing and the complexity of the induced rule set, the authors in [40] applied a wide range of approaches. The first step was to apply a feature

subset selection algorithm for reducing the dimensionality of the input set that would be given to the inducer. By doing this, the induced model can be generated in less amount of time and usually is characterized by higher predictive accuracy. The next step for reducing the complexity of the induced model was to use a pruning technique that had been incorporated to the software for the ID3 algorithm. Pruning produces models (trees) of smaller size that can avoid over-fitting, and as a result, have a higher accuracy in unknown cases.

In order to increase the quality of the induced model even more, the authors use one more step in their methodology where production rules are generated by the decision tree and they are polished, while the rules so produced are evaluated as a collection. In particular, in the first step each production rule is examined in order to decide on whether the rule should be generalized by dropping conditions from its left hand side. This is done by using contingency tables. The second and last phase checks how well the rules will function as a set. Each one of the rules is tested individually by counting the misclassifications generated by the remaining set of rules. If this number is smaller than the number of the original rule set, then the currently checked rule is removed from the final rule set. A similar approach was used in [7] in order to find a group of parameters that achieve a good trade-off between classification accuracy and the complexity of the matching model. The first step in the proposed scheme consists of defining a trade-off function that trades off model complexity for classification accuracy. In order to define the tradeoff function, the authors introduce a model complexity parameter called the *complexity ratio*. The complexity ratio is defined as the percentage contribution in overall complexity due to a parameter or group of parameters. Let $\mathcal{P}$ denote the parameter space. For any tree T, let $\mathcal{P}(T)$ denote the parameter set present in T. For a given tree T, the tree complexity function is defined as $C(T)$ as $C(T) = \sum_{p \in \mathcal{P}(T)} \text{ComplexityRatio}(p)$. $C(T)$ is simply the sum of the complexity ratios of all parameters in $\mathcal{P}(T)$. Let $S(T)$ denote the misclassification rate of tree T. The tradeoff function used is $J(T) = C(T) - S(T)$. Since both $C(T)$ and $S(T)$ are scale free, subtracting them makes sense. Now, given two trees, a starting tree T and a reduced tree (i.e., a tree with some parameters dropped), T^*, the difference in the tradeoff function simply is: $\Delta(J; T, T^*) = J(T) - J(T^*) = [C(T) - C(T^*)] - [S(T) - S(T^*)]$. The reduced tree T^* represents an improvement over the tree T if $\Delta(J; T, T^*)$ is large. The pruning process starts off with the full tree (i.e., with all parameters included) and then attempts to move to a reduced tree by dropping parameters one at a time so that the trade-off function is minimized.

8.7 Data Cleaning Language

AJAX [16] is a prototype system that provides a declarative language for specifying data cleaning programs, which consists of SQL statements enhanced with a set of primitives to express various cleaning transformations. AJAX provides a framework wherein the logic of a data cleaning program is modeled as a directed graph of data transformations starting from some input source data. Four types of data transformations are provided to the user of the system. The mapping transformation standardizes data, the matching transformation finds pairs of records that most probably refer to the same real object, the clustering transformation groups together matching pairs with a high similarity value, and finally the merging transformation collapses each individual cluster into a tuple of the resulting data source.

9 Record Matching Tools

Over the past several years, a wide range of tools for cleaning data has appeared on the market. These tools help organizations correct their database flaws in a way that is less costly and less painful than manual mending. The tools include programs that correct a specific type of error, or clean up a full spectrum of commonly found database flaws.

There are two categories of data scrubbing tools. One category includes general-purpose or development environment-oriented products like the Integrity Data Reengineering tool by Vality Technology Inc. which are more expensive and more difficult to use but more powerful, and the other category includes limited-purpose or out-of-the-box scrubbing tools like the Name and Address Data Integrity Software (NADIS) from Group 1 Software , which are cheaper and of limited scope than their general-purpose counterparts. Many products also emerged from the widespread importance of mailing lists, and they have attracted vendors with a special focus on the unique and ubiquitous needs related to mailing-list cleanup.

The "Integrity" tool by Vality automates a four-phase process that applies lexical analysis, pattern processing, statistical matching and data streaming technologies, according to an organization's specific business rules. As a first step, Integrity investigates the prior quality of the data by performing activities like data typing, metadata mining, data parsing and entity/attribute extraction and frequent analysis. "Integrity" provides conversion routines to transform the data utilizing provided business rules or custom made rules. Standardizing activities include data value correction, parsing and pattern handling, customization for special data conditions, and the integration of rule sets and reference tables. The system also identifies and consolidates records through a unique statistical matching technique that works even in the absence of common keys. One more important feature of this tool is the validation and correction of names and addresses and the appending of geographic or postal coding in real time.

Another tool that possesses extraction and transformation capabilities as well as data cleansing capabilities is Apertus Technologies Inc.'s Enterprise Integrator. Enterprise Integrator supports relational, hierarchical, network and object data models. The tool uses the Object Query Language (OQL), a declarative, SQL92-based rule specification language. OQL is used to write all transformation rules; the language contains a rich set of data conversion functions. Validation rules are used in the Enterprise/Integrator model in order to assess the quality of the data. System defined functions and tables are part of the validation procedure. Matching of different records across heterogeneous source systems is accomplished by using fuzzy logic that is fine tuned by the user. Fuzzy logic finds the closest matches to the requested information, even if no exact matches exist. Enterprise/Integrator provides the user with the capability of specifying rules for merging the matched set of records into a consolidated view.

The "Trillium" software system from Harte-Hanks Data Technologies consists of four components: a converter, a parser, a geocoder and a matcher. The converter provides a suite of tools that allow to parse, cleanse and perform data filtering, discovery and transformation tasks on any data structure. Functions provided by the converter module include: frequency analysis, word counts, custom recodes, domain range checks, pattern analysis conversions, numeric to character and length modifications. The parser module normalizes and corrects the data by performing context sensitive processing that reformats and verifies a variety

of data elements. The parser also identifies and transforms multiple names, descriptive phrases and foreign addresses into usable business data. The geocoder module, validates, corrects and enhances address and geographic information on a worldwide basis. The matcher module provides an accurate process of identifying and linking name and address records, transactions and other types of data elements.

NADIS by Group 1 Software Inc. uses expert system technology to process customer data. NADIS, which stands for Name and Address Data Integrity Software, is comprised of three products: ScrubMaster, SearchMaster, and Onlooker. ScrubMaster structures and standardizes names and addresses so that you can manage and model your data. This is accomplished by identifying every element in a name or address file and converting it to the Universal Name and Address data standard. This provides a quantitative measure of name and address integrity. SearchMaster takes the output of ScrubMaster and analyzes the relationships that exist in the data. Onlooker cleans and processes these name and address transactions and can be called from other applications such a an order-entry system.

Wizrule by WizSoft is a rules discovery tool that searches through every record in a table determines the rules that are associated with it. The user can determine which fields are examined. Wizrule then identifies records that deviate from the established rules. Wizrule supports ASCII files and any ODBC-compliant database, so it works well as a sampling tool to get a quick diagnosis on smaller projects. Wizrule permits immediate access to deviated records, allowing for discrepancies in the source data to be reconciled on the fly.

10 Future Directions and Conclusions

In this study, we have presented a comprehensive survey of the existing techniques used for reconciling data inconsistencies in database records. The techniques presented provide enough evidence of the various challenges that need to be addressed in order for users to feed data intensive applications with clean data. As database systems are becoming more and more commonplace even for non-traditional applications like data warehouses, video database systems, multi-media and spatio-temporal databases, and organizations tend to rely on their data for making mission critical decisions, it looks reasonable that cleaning or scrubbing data by matching records is going to be the cornerstone of masking errors in systems which are accumulating vast amounts of errors on a daily basis.

Based on the information that we provided, we believe that there is a lot of progress that still needs to be done. First of all, there is a big need for general purpose or domain dependent parsing software which will be able to parse any kind of records in a context sensitive manner. Making such a promise though, requires a large repository of highly available reference data. Reference data is important not only for parsing data, but also for imputing information and merging data records. Generalizing and adding features to existing matching software, that improves its effectiveness when applied to global data, is also a basic requirement in order to elevate existing record matching systems and methodologies from the realm of demonstration projects to the realm of widespread use by the government and business sectors.

Most of the record matching systems available today, offer in one way or another an algorithmic approach for speeding up the searching process and a knowledge based approach for matching pairs of records. Improvements in these processes are also essential in order to

increase the precision and the recall of the linking process as a whole. The variant nature of the matching process requires an adaptive way for on-line learning (i.e., new weights) or knowledge revision of some sort. In this way, a background process should monitor the master data and the incoming data or the various data sources that need to be merged or matched, and decide based on the observed errors, whether a revision of the knowledge base is necessary or not. Another aspect of this challenge is to develop methods that permit one to derive from the distributions of discriminating powers, inherent in particular kinds of identifying information, the proportions of errors expected in record matching projects. For the same reason, a repository of benchmark data sources with known and diverse characteristics should be made available to developers in order for them to be able to evaluate their systems. Along with benchmark and evaluation data, various systems need some form of training data to produce the initial matching model. Unless such data is readily available from previous studies, human effort is required to validate the training data. To the best of our knowledge, there are no any techniques proposed for this reason. A more extensive discussion upon future directions can be found in [44].

References

[1] Wendy Alvey and Bettye Jamerson, *Record Linkage Techniques – 1997*, Proceedings of an International Workshop and Exposition, March 1997, Federal Committee on Statistical Methodology, Office of Management and Budget.

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

[3] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[4] Abhirup Chaterjee and Arie Segev, *Data Manipulation in Heterogeneous Databases*, SIGMOD Record **20** (1991), no. 4, 64–68.

[5] ———, *Rule Based Joins in Heterogeneous Databases*, Decision Support Systems **13** (1995), 313–333.

[6] Peter Cheeseman and John Sturz, *Bayesian Classification (AutoClass): Theory and Results*, ch. 6, pp. 153–180, AAAI Press/The MIT Press, 1996.

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

[8] A.P. Dempster, N.M. Laird, and D.B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society **39** (1977), no. 1, 1–38, Series B (Methodological).

[9] Bruce D. Despain, *Probabilistic Record Linkage in Genealogy*, http://www.burgoyne.com/pages/bdespain/problink/prlgtoc.htm, August 2000.

[10] David J. DeWitt, Jeffrey F. Naughton, and Donovan A. Scneider, *An Evaluation of Non-Equijoin Algorithms*, Proceedings of the Seventeenth International Conference on Very Large Databases, 1991, pp. 443–452.

[11] Debabrata Dey and Vijay Mookerjee, *A Sequential Technique for Efficient Record Linkage*, Submitted to Operations Research Journal, 2000.

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

[13] Elke Rundensteiner (Ed.), *Special Issue on Data Transformation*, IEEE Data Engineering Bulletin, March 1999.

[14] Sunita Sarawagi (Ed.), *Special Issue on Data Cleaning*, IEEE Data Engineering Bulletin, December 2000.

[15] I. P. Fellegi and A. B. Sunter, *A Theory For Record Linkage*, Journal of the American Statistical Association **64** (1969), no. 328, 1183–1210.

[16] Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon, *AJAX: An Extensible Data Cleaning Tool*, Proceedings of the ACM SIGMOD Conference, 2000, p. 590.

[17] Mauricio Antonio Harnández-Sherrington, *A Generalization of Band Joins and the Merge/Purge Problem*, Ph.D. thesis, Department of Computer Sciences, Columbia University, 1996.

[18] Mauricio A. Hernadez and Salvatore J. Stolfo, *The Merge/Purge for Large Databases*, Proceedings of the SIGMOD 95 Conference, 1995, pp. 127–138.

[19] _____, *Real-World Data is Dirty: Data Cleansing and the Merge/Purge Problem*, Data Mining and Knowledge Discovery **2** (1998), no. 1, 9–37.

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

[21] Matthew A. Jaro, *Advances in Record-Linkage Methodology as Applied to Matching the* 1985 *Census of Tampa, Florida*, Journal of the American Statistical Association **84** (1989), no. 406, 414–420.

[22] William Kent, *The Entity Join*, Proceedings of the Fifth International Conference on Very Large Databases, 1979, pp. 232–238.

[23] Won Kim and Jungyun Seo, *Classifying Schematic and Data Heterogeneity in Multidatabase Systems*, Computer (1991), 12–18.

[24] Beth Kliss and Wendy Alvey, *Record Linkage Techniques – 1985*, Proceedings of the Workshop on Exact Matching Methodologies, May 1985, Department of the Treasury, Internal Revenue Service, Statistics Income Division.

[25] Karen Kukich, *Techniques for Automatically Correcting Words in Text*, ACM Computing Surveys **24** (1992), no. 4, 377–439.

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

[27] Priti Mishra and Margaret H. Eich, *Join Processing in Relational Databases*, ACM Computing Surveys **24** (1992), no. 1, 63–113.

[28] Alvaro E. Monge and Charles P. Elkan, *The Field Matching Problem: Algorithms and Applications*, Second International Conference on Knowledge Discovery and Data Mining (1996), 267–270, AAAI Press.

[29] _____, *An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records*, Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997, pp. 23–29.

[30] H.B. Newcombe and J.M. Kennedy, *Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information*, Communications of the ACM **5** (1962), 563–566.

[31] H.B. Newcombe, J.M. Kennedy, S.J. Axford, and A.P. James, *Automatic Linkage of Vital Records*, Science **130** (1959), no. 3381, 954–959.

[32] Howard B. Newcombe, *Record Linking: The Design of Efficient Systems for Linking Records into Individual and Family Histories*, Americal Journal of Human Genetics **19** (1967), no. 3.

[33] ———, *Handbook of Record Linkage*, Oxford University Press, 1988.

[34] Jr. N.S. D' Andrea Du Bois, *A Solution to the Problem of Linking Multivariate Documents*, Journal of the American Statistical Association **64** (1969), no. 325, 163–174.

[35] Jose C. Pinheiro and Don X. Sun, *Methods for Linking and Mining Heterogeneous Databases*, Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 1998, pp. 309–313.

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

[37] Michael Siegel and Stuart E. Madnick, *A Metadata Approach to Resolving Semantic Conflicts*, Proceedings of the Seventeenth International Conference on Very Large Databases, 1991, pp. 133–145.

[38] Benjamin J. Tepping, *A Model for Optimum Linkage of Records*, Journal of the American Statistical Association **63** (1968), 1321–1332.

[39] Vassilios S. Verykios, Mohamed G. Elfeky, Ahmed K. Elmagarmid, Munir Cochinwala, and Sid Dalal, *On the Accuracy and Completeness of the Record Matching Process*, 2000 Information Quality Conference (2000), 54–69.

[40] Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elias N. Houstis, *Automating the Approximate Record Matching Process*, Information Science **126** (2000), no. 1-4, 83–98.

[41] Vassilios S. Verykios and George V. Moustakides, *A Cost Optimal Decision Model for Record Matching*, Workshop on Data Quality: Challenges for Computer Science and Statistics (2001).

[42] Y. Richard Wang and Stuart E. Madnick, *The Inter-Database Instance Identification Problem in Integrating Autonomous Systems*, Proceedings of the Fifth International Conference on Data Engineering, 1989, pp. 46–55.

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

[44] William E. Winkler, *The State of Record Linkage and Current Research Problems*, Proceedings of the Survey Methods Section (1999), 73–79.