A Knowledge-Based Approach for Duplicate Elimination in Data Cleaning

Wai Lup Low, Mong Li Lee and Tok Wang Ling

School of Computing, National University of Singapore,3 Science Drive 2, Singapore 117543

Abstract

Existing duplicate elimination methods for data cleaning work on the basis of computing the degree of similarity between nearby records in a sorted database. High recall can be achieved by accepting records with low degrees of similarity as duplicates, at the cost of lower precision. High precision can be achieved analogously at the cost of lower recall. This is the *recall-precision dilemma*. We develop a generic knowledge-based framework for effective data cleaning that can implement any existing data cleaning strategies and more. We propose a new method for computing transitive closure under uncertainty for dealing with the merging of groups of inexact duplicate records and explain why small changes to window sizes has little effect on the results of the Sorted Neighbourhood Method. Experimental study with two real world datasets show that this approach can accurately identify duplicates and anomalies with high recall and precision, thus effectively resolving the recall-precision dilemma.

Key words: Data cleaning, duplicate elimination, knowledge-based system

1 Introduction

The amount of data organizations are handling today is increasing at an explosive rate. Dirty data is pervasive and organizations are faced with an insurmountable task of maintaining correct and consistent data in these large databases. Clean data is of critical importance for many industries over a wide variety of applications [Kim96], including marketing communications, commercial householding, customer matching, merging information systems, medical records etc. [Lim98]. Given the 'garbage in, garbage out' principle, dirty data will not be able to provide data miners with correct information. It is difficult for managers to make logical and well-informed decisions based on information derived from such data. [SSU96] identified data cleaning as one of the database research opportunities for data warehousing into the 21st century.

Data cleaning is often studied in association with data warehousing, data mining and database integration. These areas have received much attention from the database research community in recent years. However, research focus of the past has been on the problem of view integration (or sometimes referred to in literature as schema integration), where the aim is the reconciliation of differences at the conceptual level. View integration has been investigated in numerous comprehensive studies on integration frameworks [CdGL$^+$99,MKCWB97], mediator systems [GMPQ$^+$97,BGF$^+$97] and schematic conflict resolution [BLN86,LL97]. Data cleaning, which is the next logical step that involves reconciliation of data at the instance level, has received little attention. Research in data cleaning is orthogonal to the work on view integration and our work assumes that the conflicts at the conceptual/schema level have been resolved and a global reconciled schema has been obtained. Data reconciliation at the instance level faces totally different challenges and problems.

The problem of dirty data was recognized way back in the 1950s when census work was being carried out [KA85]. There are many causes to dirty data: misuse of abbreviations, data entry mistakes, embedding control information in data (e.g. printer control commands in data fields for formatting output), different phrases (ASAP versus 'at first chance'), duplicate records, missing values, localization differences (the use of a student's aggregate point versus lecturers' progress reports as a measure of performance by different departments of an university), spelling variations, unit differences, outdated codes, etc. [Lim98]. Data cleaning refers to a series of processes by which the quality of data is improved by dealing with the problems listed above. Notably, data cleaning processes will perform format standardization, anomaly removal, error correction and duplicate elimination. Data cleaning takes on many forms, and it is very complex, time-consuming and expensive. A simple data cleaning process would be to remove exact duplicate records from databases. More complex processes take into consideration inexact duplicate records, inferencing of missing values and error correction. In the extreme case, data cleaning may involve re-engineering of the entire data flow, from point-of-capture to database design, typically affecting all data applications in the organization.

Recent research efforts have focused on the issue of duplicate elimination in databases. This entails trying to match inexact duplicate records, which are records that refer to the same real world entity while not being syntactically equivalent. However, existing strategies suffer from some common problems : a lack of generic knowledge management schemes for data cleaning, difficulty in achieving high recall and precision at the same time and the loss of precision from the computation of transitive closure. In this paper, we will focus on the data cleaning strategies to be used on existing and legacy textual databases, with emphasis on their applicability to a wide variety of datasets and ability to deal with diverse inconsistencies in data. The metrics used to benchmark strategies are recall, precision and speed. The contributions of this paper are:

(1) Introducing a knowledge-based framework for effective representation and management of human expertise for data cleaning.
(2) Resolving the recall-precision dilemma for duplicate elimination.
(3) Reducing the loss of precision from transitive closure computation by the introduction of uncertainty into the computation.

The rest of the paper is organized as follows. Section 2 gives the metrics used and challenges faced in data cleaning. Section 3 surveys related research in this area. Section 4 details our proposed knowledge-based approach and how it can overcome the problems plaguing existing methods. Section 5 describes the implementation of the system and results of our experiments. Finally, Section 6 summarizes the contributions of this paper and highlights some directions future work in this area can take.

2 Preliminaries

We review here some general concepts that are important for a full understanding of the proposed strategies discussed in the following sections. We also highlight some of the main difficulties when designing data cleaning strategies.

2.1 A Conceptual Model

A high level conceptual data cleaning model is presented in Figure 1. First, we have a "dirty" dataset with a variety of errors and anomalies. A list of common causes of dirty data is described in [Mos98]. Cleaning strategies are applied to the dataset with the objective to obtain consistent and correct data as the output. The effectiveness of the cleaning strategies will thus be the degree by which data quality is improved through cleaning.

2.2 Benchmarking Effectiveness

We define two metrics that benchmark the effectiveness of data cleaning strategies :

(1) Recall.
This is also known as *percentage hits*, *true positives* or *true merges* in some literature. It is defined as the percentage of duplicate records being correctly identified. Assume we have 7 records $A_1, A_2, A_3, B_1, B_2, B_3, C_1$, with $\{A_1,\ A_2,\ A_3\}$ and $\{B_1,\ B_2,\ B_3\}$ being different representations of records A and B respectively. If a cleaning process identifies $\{A_1, A_2, C_1\}$ and $\{B_1, B_2\}$ as duplicates, then we have a recall of $\frac{4}{6} \times 100\% = 66.7\%$ as it identified 4 $(A_1,\ A_2,\ B_1,\ B_2)$ out of 6 duplicates.

(2) False-Positive Error.
This metric is the antithesis of the *precision* measure of a method. It is sometimes referred to as *false merges* in literature. This is defined as the percentage of records wrongly identified as duplicates, i.e.

$$\text{False-Positive Error} = \frac{\text{no. of wrongly identified duplicates}}{\text{total no. of identified duplicates}} \times 100\%$$

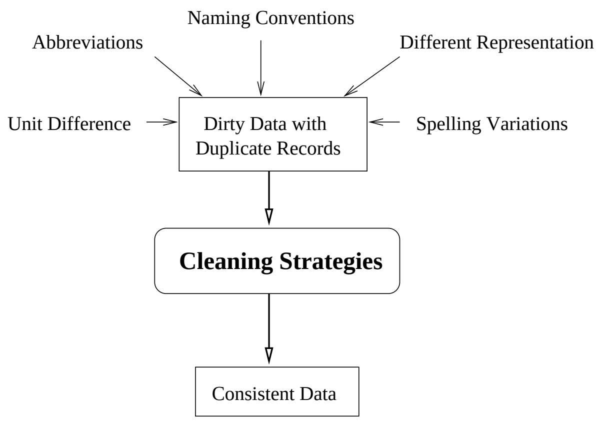


Fig. 1. An Overall Conceptual Model

An example of 2 similar records

Name	Address	Sex	Tel. No.
Tan Ah Kow	Blk 555, Bukit Merah, #11-01, S(112555)	M	222 1256
A. K. Tan	Apt Blk 555, B. Merah #11-01, Singapore 112555	M	2221256

and

$$\text{Precision} = 100\% \text{ - False-Positive Error.}$$

In the above example, the process incorrectly identified C_1 as a duplicate record and will have a false-positive error of $\frac{1}{5} \times 100\% = 20\%$. Precision is thus 80% (100% - 20%).

2.3 Problems and Difficulties

We have noted in Section 1 that data cleaning consists of a series of complex processes. We highlight here the main problems faced during data cleaning which increase its complexity:

(1) Uncertainty and Risk.
In real-life datasets, one can rarely be sure that two inexact records represent the same entity, even though they might be very similar (they might even share the same key!). Hence, uncertainty is introduced into the decision making process. We cannot say with absolute certainty that the two records in Table 1 represent the same person without further information or checks, even though they look very much the same. While there is a number of uncertainty algebras, it is unclear how they can be applied in this domain. There is much future work to do in the exploration of domain applicability of uncertainty algebras and automated uncertainty calculations. [Wie93]

(2) Verification of Results.
Metrics defined in Section 2.2 are not easily obtained when cleaning real-world databases, which often contain tens of thousands or even millions of records. This brings about the problem of finding out the true number of duplicate records and errors in the data to obtain metrics that benchmark strategies. This will entail human inspection over the whole database, which is infeasible in some cases. Even when human inspection is performed, it is difficult sometimes even for trained staff to ascertain some duplicate records and errors.

(3) Database Dependence.
Benchmarking new algorithms and methods of data cleaning proved to be difficult as the performance is heavily dependent on the database, with different attribute and error types in the data. A method performing very well on one database might perform very badly on another. Coming up with a representative database is not feasible as no single database can satisfactorily model the characteristics of all real-world datasets.

3 Related Works

There have been numerous contributions in this line of research, each focusing on different issues. We review existing data cleaning methods in this section.

3.1 Pre-processing of Dirty Data

This refers to the processes prior to starting the data cleaning process, with efforts to scrub the data into a more consistent state. Pre-processing dirty data prior to the data cleaning process leads to more standardized and consistent data going into the merge/purge process and better results can be achieved [LLLK99]. The reason for better results is the increase in likelihood of potential duplicates being close together after sorting is performed on the data due to increased similarity. Pre-processing usually involves external reference files, abbreviation standards and automatic spelling correction. [LLLK99] also proposed that the equivalence of records can be determined by viewing their similarity at three levels : token, field and record levels.

3.2 Sorted Neighbourhood Method

The standard method for detecting exact duplicates in a database is to sort the database and then check if the neighbouring records are identical [BD83]. In order to detect inexact duplicates in a database, the most reliable way is to compare every record with every other record, which is an $O(N^2)$ operation. [HS95] proposed a *Sorted Neighbourhood Method (SNM)* to detect approximate duplicates which requires only wN comparisons, where w is the size of the sliding window.

The SNM consists of three steps :

(1) Create Keys. A key is computed for each record in the database by extracting relevant fields or portions of fields which form an important discriminating attribute. The choice of the key depends upon an "error model" that draws from domain-knowledge. The key selection process is a highly knowledge-intensive and domain-specific process [HS98], as the "key designer" has to know the characteristics of the data (and the characteristics of the errors in the data), before being able to choose a good key. Domain-dependent heuristics can be used to guide the selection of the key.
(2) Sort Data. Sort the records in the database using the key computed in Step 1.
(3) Merge. Move a fixed size window through the sequential list of records limiting the comparisons for matching records to those records in the window. If the size of the window is w records, then every new record entering the window is compared with the previous $w-1$ records to find "matching" records. The first record in the window slides out of the window as shown in Figure 2.

The drawbacks of the SNM lie in the fact that it depends heavily on the proximity of duplicate records after sorting, which in turn, depends on the key chosen for sorting. If duplicate records are far apart after sorting, it is unlikely that they will appear in the same window during the scanning process, and hence, will be missed. The obvious solution is to increase the window size for the scan, but this will lead to increased main memory requirements and computational complexity as more comparisons are needed.

The *Duplicate Elimination SNM* (DE-SNM) [Her96] improves the results of SNM by first sorting the records on a chosen key and then dividing the sorted records into two lists : a duplicate list and a no-duplicate list. The duplicate list contains all records with exact duplicate keys. All the other records are put into the no-duplicate list. A small window scan is first performed on the duplicate list to find the lists of matched and unmatched records. The

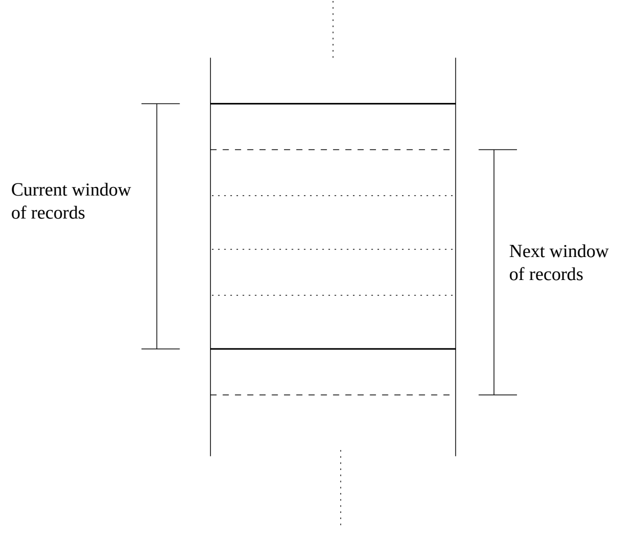


Fig. 2. Sliding Window Scan

list of unmatched records is merged with the original no-duplicate list and a second window scan is performed. However, the drawbacks of the SNM still persists in DE-SNM.

Instead of using a window scan, data may be partitioned into clusters such that matching records appear in each cluster. Merge/purge processes can then be applied to each cluster in parallel. However, we note that if the proportion of duplicates in the database is small (usually the case), this procedure will result in many singleton clusters.

3.3 Priority Queue Algorithm

As an alternative to the SNM, [Alv97] suggests using the priority queue algorithm which uses a priority queue of sets of records belonging 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 represented in a priority queue. 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. If one of these comparisons succeeds, then the record belongs in this cluster. 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. If this action causes the size of the priority queue to exceed its limit, then the lowest priority set is removed. The algorithm is explained in detail in [Alv97]. However, this method still faces the “window size” problem of the SNM when determining the size of the queue. Moreover, heuristics need to be developed for choosing the representative records in the priority queue, which will affect the results greatly.

3.4 Multi-pass Data Cleaning Methods

Single-pass algorithms go through the *create key, sort, merge* cycle once. Multi-pass algorithms refer to those that goes through the cycle several times, each sorting on a different key. This method assumes that no single key is unique enough to bring all inexact duplicates together, and by sorting on different keys, the chances of missed duplicates become lower. Comparatively, single-pass algorithms' emphasis is on creating a "good" key and the duplicate identification process can be more complicated since this cycle is only done once.

For multi-pass algorithms, the transitive closure is computed over the identified duplicates [HS95],[Alv97]. This is because if A is equivalent to B, and B is equivalent to C, then logically speaking, A is also equivalent to C under the assumption of transitivity. Computing the transitive closure thus, will group "similar records" which may be duplicates. However, computing the transitive closure in such a way is likely to increase the false-positive error. This problem will be discussed in Section 4.4. Comparatively, most experimental results show that several "cheap" passes followed by the computation the transitive closure over the identified duplicates perform better than a single "expensive" pass.

3.5 Incremental Data Cleaning

The Basic Incremental Merge/Purge Procedure (BIMP) and Increment Sampling Incremental Merge/Purge Procedure (ISIMP) [Wal98] avoid re-processing data that has previously been processed when increments of data need to be merged with previously processed data. This involves storing previously gathered information and the selection of "prime-representative" records which is representative of the whole original database. The data cleaning process then uses only the representative records and the data increments. The savings are potentially tremendous if the number of prime representatives is very much smaller than the number of records. The main problem with these methods lie in the difficulty of choosing the prime representatives. The ability of chosen sample records to represent the whole dataset is also in suspect.

3.6 Cleaning with Domain Knowledge

It is also proposed that equivalence of records are defined by a set of equational axioms, which is termed equational theory [HS95] [Her96] [HS98]. Determining record equivalence requires much more information than simply their sort keys and [HS95] considers this an complex inferential process. A high level declarative rule language that can express the equivalent logic will be useful in a general merge/purge application.

Different instances of the same entity in different databases can be identified by making inferences using other shared attributes in more complicated cases where there are no common key identifiers and are different in representation [WM89]. An example given in [WM89] is shown below :

```
(Professor) :   Do you know who TK Wong is?
(TA)        :   No. Does he come to the morning class?
(Professor) :   Yes, when he comes at all.
(TA)        :   How well is he doing in the class?
(Professor) :   Not well.  He's always falling asleep.
(TA)        :   Is he quiet?
(Professor) :   No! He keeps complaining about our LISP compiler.
(TA)        :   Oh, sure! I call that guy Big Mouth.
```

So, even though no common unique key is used to identify the student, inferencing can be used to eliminate all other possibilities. This inferencing process is possible with the use of other attributes (class, track record) and domain knowledge. Handling data in such situations is made difficult by the uncertainty and risk involved [Wie93], especially in cases where data are from heterogeneous and diverse sources. We note that in such situations, these other attributes will form a concatenated alternate key if they can always uniquely identify a student.

3.7 Domain-Independent Data Cleaning

There has also been work on domain-independent techniques that require no domain knowledge as input to the data cleaning process. Generally, these techniques assume that records have the same high level schema and are made up of alphanumeric characters. [Alv97] presents a domain-independent algorithm for detecting approximately duplicate database records. It makes use of several domain-independent algorithms outlined in [ME96] and works on textual databases. These methods typically treat database records as strings and compute record similarity using string comparison algorithms.

WHIRL [Coh98] is a logic for reasoning about the similarity of names using the vector space model. Although the author cites its use in the integration of databases that lack common domains, we believe it can be used as a domain-independent technique for duplicate identification in the data cleaning process.

Potter's Wheel [RH00] is an interactive framework which offers graphical specification of data transformations through a spreadsheet-like interface. It allows the user to try various transformations interactively, observe their effects on the data, and undo them if they are inappropriate. The graphical interface for specifying data transformations might not be adequate for specifying transformations involving complicated business logic.

3.8 Integrating Data Cleaning with Database Management Systems

Data cleaning can leverage on the years of research gone into database management systems (DBMS) by integrating data cleaning processes as SQL extensions. Concurrent to our work, [HGS99] proposes a framework which uses an SQL extension for specifying data cleaning operators, which include data transformation, duplicate elimination and multi-table matching. [HGS99] also investigates the optimization techniques that can be used to optimize the data cleaning process. Having a SQL-like language extension has the advantage of increased usability if users are familiar with SQL. However, optimization remains a big challenge in this framework, especially when the matching operator used in the framework relies on a Cartesian product-based algorithm. Performance will suffer during the processing of large databases as a result.

4 Knowledge-Based Approach for Duplicate Elimination

In this section, we present a knowledge-based approach for intelligent data de-duplication and cleaning. This approach incorporates a framework that provides a complete strategy for

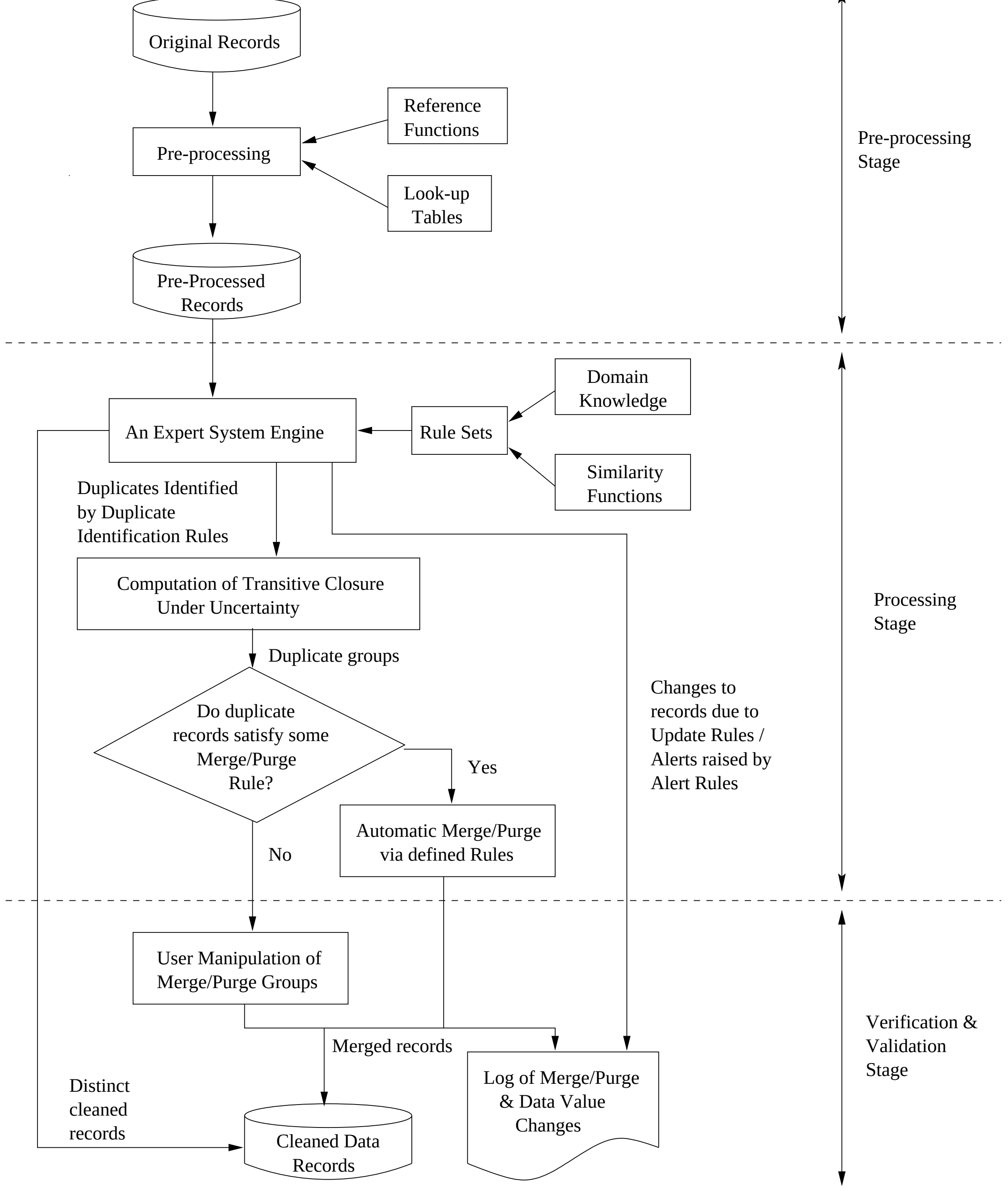

Original Records
Reference Functions
Pre-processing
Look-up Tables
Pre-Processed Records
Pre-processing Stage
Domain Knowledge
An Expert System Engine
Rule Sets
Similarity Functions
Duplicates Identified by Duplicate Identification Rules
Computation of Transitive Closure Under Uncertainty
Processing Stage
Duplicate groups
Do duplicate records satisfy some Merge/Purge Rule?
Changes to records due to Update Rules / Alerts raised by Alert Rules
Yes
Automatic Merge/Purge via defined Rules
No
User Manipulation of Merge/Purge Groups
Verification & Validation Stage
Merged records
Distinct cleaned records
Log of Merge/Purge & Data Value Changes
Cleaned Data Records

Fig. 3. A Knowledge-Based Framework for Data Cleaning

standardizing, anomaly detection and removal, and duplicate elimination in dirty databases. We discuss the algorithms and methods used in the framework and highlight the benefits of our knowledge-based approach.

4.1 A Framework for Data Cleaning

Figure 3 shows our proposed framework. It consists of three stages :

(1) Pre-processing Stage.
The data records are first conditioned and scrubbed of any anomalies we can detect and

correct at this stage. Data type checks and format standardization can be performed (e.g. 1 Jan 2000, 01/1/2000, and 1st January 2000 can be standardized to one fixed format). Inconsistent abbreviations used in the data can also be resolved at this stage (e.g. All occurrences of 'Rd.' and 'Rd' in the address field will be replaced by 'Road'. Occurrences of '1' and 'A' in the sex field will be replaced by 'Male', and occurrences of '2' and 'B' will be replaced by 'Female'.)

The conditioning and scrubbing of the records can be done via reference functions and look-up tables. For example, a function can take in a date input and output it in a specified format for the purpose of standardizing date representations. A two-column look-up table might contain an abbreviation and its standardized equivalent.

The output of this stage will be a set of conditioned records which will be input to the processing stage.

(2) Processing Stage.

The conditioned records are next fed into an expert system engine together with a set of rules. Each rule will fall into one of the following categories :

(a) Duplicate Identification Rules. These rules specify the conditions and criteria for two records to be classified as duplicates. Functions to compare text similarity, procedures to perform string manipulation, and complex logic for determining record equivalence can be coded into or referenced by the rules. The pseudocode of one of the rules used in the experiments (detailed in Section 5) is produced below.

```
DEFINE RULE COMPANY_RULE
  INPUT RECORDS : A B
  IF
      (A.currency == B.currency)          AND
      (A.telephone == B.telephone)        AND
      (A.telephone != EMPTY_STRING)       AND
      (SUBSTRING_ANY(A.code,B.code) == TRUE) AND
      (FIELDSIMILARITY(A.address,B.address) > 0.85)
  THEN
      DUPLICATES (A B), CERTAINTY=0.85
      UPDATE_LOGS
```

For the above rule to be activated, the corresponding currencies and telephone numbers must match. Telephone numbers must also not be empty, and one of the codes must be a substring of the other. The addresses must also be very similar (Similarity of addresses using Function 'FieldSimilarity' must be higher than 0.85). Records classified as duplicates by this rule has a Certainty Factor of 85%.

(b) Merge/Purge Rules. These rules specify how duplicate records are to be handled. A simple rule might specify that only the record with the least number of empty fields is to be kept in a group of duplicate records, and the rest be deleted. Much more complex actions can be specified in a similar fashion.

(c) Update Rules. These rules specify the way data is to be updated in a particular situation. For example, in the event that the quantity field of a book order record is empty, and the buyer has no other purchases, a rule specifies that the quantity field is to be updated with the value of 1. If the buyer has other purchases, the quantity is to be updated with the minimum quantity of all purchases belonging to the same buyer.

(d) Alert Rules. The user might want an alert to be raised when certain events occur. For example, when the database shows that a terminated employee is still receiving payments from the company, it might be correct as a case of post-dated payments or, it could indicate data errors. It might even be a case of fraud!

Alert rules can also be used for the checking and warning of constraint violations. We illustrate how this can be done using two examples. We might have a constraint that says all part-time employees must have a positive value in their hourly-wage fields. We will want to raise an alert if a part-time employee has a non-positive value in the hourly-wage field of the employee record. The rule will have the following form :

```
DEFINE RULE HOURLYWAGE_CHECK
  INPUT RECORD : A
  IF
      (A.EmployeeType == PART_TIME)  and
      (A.HourlyWage <= 0)
  THEN
      RAISE_ALERT
```

If we have a *functional dependency* constraint, $X \rightarrow Y$, it can be checked simply by a rule that says if two records have the same *X-value*, then raise an alert if they do not have the same *Y-value*. The rule might have the form :

```
DEFINE RULE FD_CHECK
  INPUT RECORDS : A B
  IF
      (A.X == B.X)  and
      (A.Y != B.Y)
  THEN
      RAISE_ALERT
```

Such rules are especially useful when the database management system in which the data resides does not support the checking of constraints, or such checking was not implemented by the database administrator. The rules can also be used in cases when integrity constraints were not considered in the database design in the first place.

All the rules will then fire in an opportunistic manner when the pre-processed records are fed into the expert system engine, whose implementation uses the Java Expert System Shell (JESS) [FH99]. The pattern matching and rule activation mechanisms of JESS makes use of the Rete algorithm [For82]. To avoid pairwise comparison for each and every record in the database (which may be infeasible in large databases), we employ the Sorted Neighbourhood Method (SNM). Hence, the rules will only act on one window of records in the expert system engine at any point in time. After the duplicate record groups are identified, we consider using a new method to compute the transitive closure under uncertainty for these groups to increase the recall. The merge/purge rules will then act on the duplicate record groups that satisfy their conditions. The Rete Algorithm is discussed in Section 4.2. JESS and rule representation are discussed in Section 4.3.

(3) Human Verification and Validation Stage.

In this stage, human intervention will be required to manipulate the duplicate record groups for which merge/purge rules are not defined. Upon human inspection, false positives can be taken out of these groups and appropriate merge/purge procedures can be carried out.

The log report provides an audit trail for all actions and the reasons for the actions made during the pre-processing and processing stages. This log can be inspected for consistency and accuracy and if necessary, wrong merges and incorrect updates can be undone. Undoing the changes can be simple if all changes and updates are made to a temporary copy of the original database. The log can also be used for validating the rule base. For example, if a rule consistently classifies the wrong records as duplicates, or updates values incorrectly, then it should probably be taken out or have its parameters changed.

4.2 The Rete Algorithm

The Rete Algorithm [For82] is an efficient method for comparing a large collection of patterns to a large collection of objects. It finds all the objects that match each pattern. The algorithm was developed for use in production system interpreters, and has been used for systems containing from a few hundred to more than a thousand patterns and objects. The Rete Algorithm has been used in several expert system shells including OPS5, its descendant ART, and CLIPS.

A basic production system checks each if-then statement to see which ones should be executed based on the facts in the database, looping back to the first rule when it has finished. The computational complexity is of the order $O(RF^P)$, where R is the number of rules, P is the average number of patterns in the condition part of the rules, and F is the number of facts in the knowledge base. This escalates dramatically as the number of patterns per rule increases. This is inefficient as most of the tests made on each cycle will have the same results as the previous iteration. The Rete Algorithm avoids unnecessary recomputation by remembering what has already been matched from cycle to cycle and then computing only the changes necessary for the newly added or newly removed facts [GR98]. As a result, the computational complexity per iteration drops to something more like $O(RFP)$, or linear to the size of the fact base [FH99]. However, saving the state of the system in this way consume considerable amounts of memory. In general, this trade-off of memory for speed is worthwhile. The reason we choose this algorithm to power our expert system engine is primarily due to its efficiency.

In the expert system engine of the proposed framework, the data records are the facts. When the records matches the "patterns" specified in the rules, they are activated or fired. The specified actions in the consequent part of the rule, which varies with the different rule types, will then be carried out.

Our discussion here on Rete Algorithm is rather brief due to space constraints. Details of this algorithm and its performance can be found in [For82] and [GR98].

4.3 Rule Representation and Effectiveness

Rules, which form the knowledge-base of the framework, are written in the Java Expert System Shell (JESS) language. JESS [FH99] is a rule engine and scripting environment written in Sun's Java language and was inspired by the CLIPS [Ril99] expert system shell. The data cleaning rules are represented as declarative rules in the JESS engine. A rule will generally be of the form :

```
if <condition>
then <action>
```

The action part of the rule will be activated or *fired* when the conditions are satisfied. We note here that complex predicates and external function references may be contained in both the condition and action parts of the rule.

Anyone with subject matter expertise will be able to understand the business logic of the data and can develop the appropriate conditions and actions, which will then form the rule set. Experiences with knowledge-based data cleaning suggested that predicate logic can be used for initial rule formulation [HP99] for easy understanding and translating rules of this representation into the target language is relatively straight-forward.

The complexity of the rule language is an important issue in such applications. Although the declarative style of the JESS language makes it intuitive, some knowledge of the working of the Rete Algorithm is required for optimizing rule efficiency. Noting the structure of the rule language, simple rules may be generated automatically by the system when supplied with necessary parameters. More efficient, flexible, complex and precise rules may be achieved through hand-coding. We note here that while the engine based on Rete Algorithm provides efficient pattern matching and rule activation mechanisms, it is the rule effectiveness that determines the recall and precision of the cleaning process.

4.4 Benefits of Knowledge-Based Approach

In this section, we discuss the advantages offered by the knowledge-based approach and see how it can overcome the short-comings of existing strategies.

(1) Knowledge Representation and Application.

Domain knowledge has been identified as one of the main ingredients for successful data cleaning [May99]. After all, what are considered duplicates or data anomalies in one case, might not be in another. Such knowledge is domain-dependent and is derived naturally from the business domain and associated knowledge. The business analyst with subject matter expertise will be able to fully understand the business logic governing the situation and can provide the appropriate knowledge to perform data cleaning.

However, the issue of knowledge management in data cleaning strategies has not been well dealt with. No representation of the knowledge used in data cleaning has been suggested and the closest works are [HS95], [Her96], [HS98], [HGS99] and [RH00]. In existing methods, domain knowledge has been restricted to the selection of similarity measures and the threshold for merge/purge processes.

Before we can propose a representation, the following questions need to be answered :

- What type of knowledge is suitable?
- How can we represent the knowledge in a form that we can use for data cleaning efficiently?
- How do we manage the knowledge?

In our framework, the domain knowledge is represented in the form of rules. The different rule types provides the various functions needed for the cleaning process. The expert system engine provides the pattern-matching and rule-firing capability in an efficient manner, thus "applying the business rules" to perform data cleaning.

We note that such rules are rather persistent in nature and need not be changed or modified often, due to the static nature of business rules from which they are derived. Changes are likely to be restricted to small modifications made to parameters to fine-tune performance. The rules can also be re-used without modification by different applications in the same domain if they possess similar business rules. This makes domain-specific rule packages feasible. For example, any data cleaning application that involves customer databases are likely to use the same rule set.

The knowledge-based approach also makes it easy to offer an *explanation facility.* Such a facility enables post-processing analysis, when each action can be explained, traced and accounted for. This is especially useful in cases where we need to know why certain values have been altered, and the reason for updating the data with this value. By keeping track on the rules that fired on this record, we can trace the reasoning and explain the changes. This facility can also be used for fine-tuning the knowledge-base in the Human Verification and Validation Stage, as explained in earlier in this section.

(2) The Recall-Precision Dilemma.

Existing methods work on the basis of computing the degree of similarity between records, and classify them as duplicates if the degree of similarity is above a defined threshold. For example, assume we have a function, $f(R_1, R_2)$ that takes in 2 records, R_1 and R_2, as arguments and returns their degree of similarity. If TH is the defined threshold, then R_1 and R_2 will be classified as duplicate records if $f(R_1, R_2) \geq TH$.

To achieve high recall, i.e. to detect more duplicate records in the dataset, we can accept records with low degrees of similarity as duplicates. This can be done by using a smaller value for the threshold. However, this will lead to lower precision, i.e. higher false-positive error, as more "duplicates" are identified wrongly. Analogously, we can achieve higher precision at the cost of lower recall, by setting a higher threshold.

Thus, it seems that using such approaches, we can only better one of these measures at the cost of the other, thereby forcing practitioners of such methods to decide on an optimal crossover-point for the measures. We term this the *recall-precision dilemma.*

Well-developed rules are effective in identifying true duplicates but are strict enough to keep out similar records which are not duplicates. Higher recall can then be achieved with more rules. Hence, an increase in the number of well-developed rules can achieve higher recall without sacrificing precision. Thus, our approach can effectively resolve the recall-precision dilemma.

(3) Loss of Precision from Computation of Transitive Closure.

[HS95] and [Alv97] have advocated that the computation of transitive closure on the pairs of identified duplicate records will increase the recall under the assumption of transitivity. This computation seems natural after the application of the multi-pass approach [HS98], which was discussed in Section 3. This allows duplicate records to be detected even without being in the same window during the SNM window scans. The idea is as follows : If records a and b are found to be similar and, at the same time, records b and c are also found to be similar, the transitive closure step can mark records a and c as similar without being in the same window at any point in time.

However, we note that this procedure can raise the false-positive error as incorrect pairs are merged due to the fact that we are dealing with *inexact equivalence.* For example, we have two groups of very similar "duplicate" words : (FATHER,MATHER) and (MATHER,MOTHER). The transitive closure step will then merge the two groups into a single group of (FATHER,MATHER,MOTHER) due to "MATHER" appearing in both groups. This is obviously wrong since "FATHER" and "MOTHER" are totally different words. Any attempt in automated merging and purging of records in this group may result in merging these two different words into one.

This problem will magnify as more groups are merged into a single group due to different common records linking various pairs, i.e. if we have the following pairs of identified duplicates,

$$(R_1, R_2), (R_2, R_3), \ldots, (R_{n-2}, R_{n-1}), (R_{n-1}, R_n) \ ,$$

the transitive closure computation for these groups will result in them merging into one large group of

$$(R_1, R_2, \ldots, R_{n-1}, R_n).$$

In this case, the first pair will probably be different records from the later pairs. This will result in many wrongly identified "duplicates", leading to low precision.

We outline a method for computing the transitive closure under uncertainty which can reduce the number of wrongly merged duplicate groups. We can attach a certainty factor, cf, to each duplicate identification rule, where $0 < cf \leq 1$. This value represents our confidence in the rule's effectiveness in identifying true duplicates. We can assign a high certainty factor to a strict rule, if we are sure that it will identify true duplicates. Record groups identified by this rule will have this certainty factor. Analogously, we assign smaller values for rules that are less strict. During the computation of the transitive closure, we compare the resultant certainty factor of the merged group against a user-defined threshold. This threshold represents how tight or confident we want the merges to be. Any merges that will result in a certainty factor less than the threshold will not be carried out. The rationale behind this method is : As more and more groups are merged during the transitive closure step, we are less and less confident that all the records in the resultant group are representations of the same real-world entity.

To support this method of computing transitive closure, the structure of the Duplicate Identification Rules thus need to be modified to :

```
if <condition>
then <records are duplicates with certainty factor cf>
```

We present 4 examples here for clarity. A and B are records from the dataset. CF and TH represents the certainty factor and threshold respectively.

(a) Example 1 : Merge Records (A B) with CF=0.8, TH=0.9

The rule that identified this duplicate group is assigned a CF of 0.8. However, it is lower than TH. Records A and B are kept unchanged.

(b) Example 2 : Merge Records (A B) with CF=0.8, TH=0.7

Records A and B will be merged as $CF > TH$.

(c) Example 3 : Merge Records (A B) with CF=0.8, (B C) with CF=0.75, TH=0.7

As the combined CF of the merged group will be $(0.8 \times 0.75) = 0.6 < TH$, the above 2 record pairs will not be merged, but remain as separate groups. However, the merging will be done within each group, i.e. (A B) and (B C) as $CF > TH$ in both cases.

(d) Example 4 : Merge Records (A B) with CF=0.9, (B C) with CF=0.85, (C D) with CF=0.8, TH=0.5

The groups (A B) and (B C) will be considered first, as these groups have higher CFs. They will be merged to form (A B C) with $CF = 0.765$ (since $0.9 \times 0.85 = 0.765$). Then, this group will be merged with (C D) to form (A B C D) with CF=0.612 (since $0.765 \times 0.8 = 0.612$), and the CF is still greater than TH. However, if $TH = 0.7$, (A B C) and (C D) will remain separate, as the resultant CF of the merged group (0.612) will be less than TH.

The main problem with this method is that a record may exist in more than one group as in Examples 3 and 4 given above. Note that the final grouping may not be unique when there are groups with the same certainty factors. On-going research is looking into how this method can be improved.

A method based on the Dempster-Shafer theory of evidence is presented in [LSS94] to model uncertainty in attribute values. A mass value is assigned to every subset of the environment, which represents the portions of belief committed to the sets. Our proposal is totally different. Our certainty refers to our belief and confidence in the effectiveness of a rule, i.e. the uncertainty of a rule, and not the uncertainty in attribute values. For

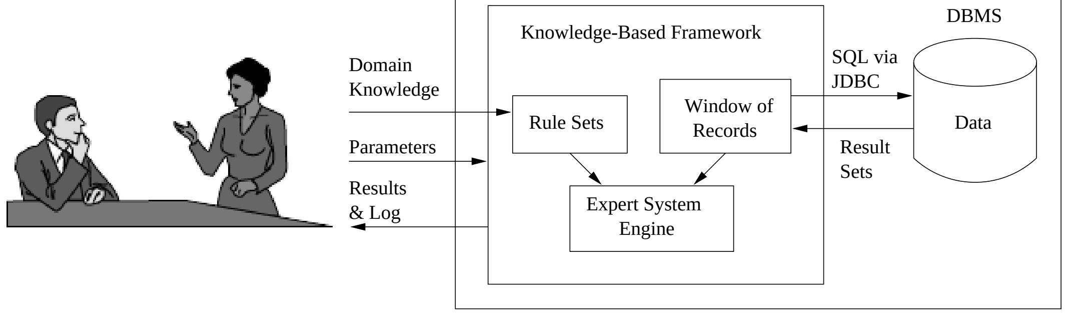


Fig. 4. Architecture of Intelligent Data Cleaning System

each rule, only one single certainty factor is required.

(4) Implementing Existing Methods Using the Framework

The powerful declarative rule language provided by JESS can implement the comparison algorithms and similarity functions used by existing methods. For example, a rule can determine record equivalence by partitioning a field into separate tokens and sorting them before performing comparisons. This effectively implements the method proposed by [LLLK99].

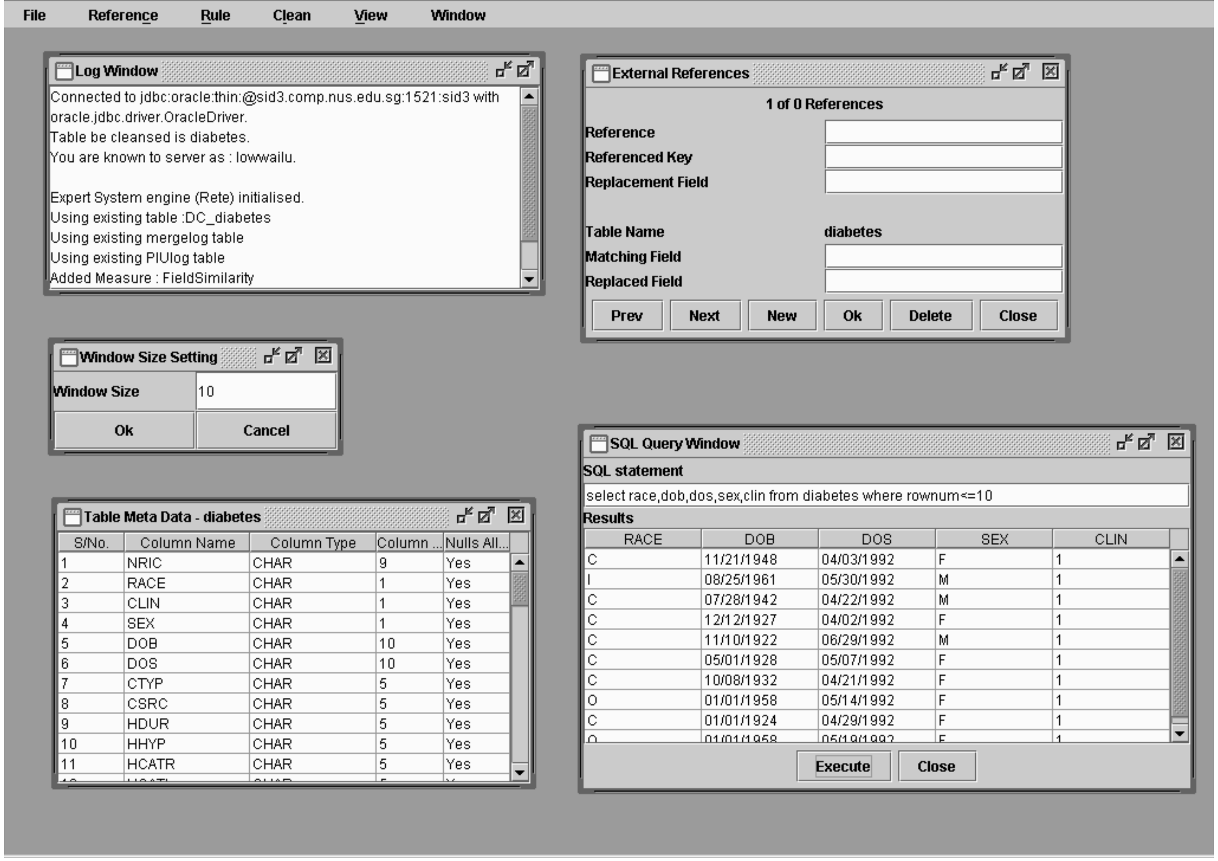
The framework can also degenerate gracefully into a domain-independent strategy by using only rules that treat records as strings and employ string comparison functions to compare these "strings". Hence, the framework can implement existing methods and current strategies form a subset of data cleaning strategies this approach offers.

5 Performance Study

In this section, we describe the implementation and performance evaluation of IntelliClean, a data cleaning system that employs the proposed knowledge-based framework for duplicate elimination.

5.1 Experimental Testbed

Based on the ideas and techniques presented in this paper, we implemented the IntelliClean system using Java 1.2 with the interface employing the use of Java Swing components. As a result, the system runs on multiple platforms. The data source connection is made using JDBC (Java Database Connectivity) and hence, the data source can be located at any local or remote relational database servers. For our experiments, the database server is a SUN Enterprise 450 server located in a remote machine connected via a local area network. The database management system used is Oracle 8. The client computer running IntelliClean is a Pentium 200 MMX system with 64 MB RAM running Windows NT 4.0. The architecture of the implementation is depicted in Figure 4. A screen capture of IntelliClean is shown in Figure 5.



Log Window

```
Connected to jdbc:oracle:thin:@sid3.comp.nus.edu.sg:1521:sid3 with
oracle.jdbc.driver.OracleDriver.
Table be cleansed is diabetes.
You are known to server as : lowwailu.

Expert System engine (Rete) initialised.
Using existing table :DC_diabetes
Using existing mergelog table
Using existing PIUlog table
Added Measure : FieldSimilarity
```

External References

1 of 0 References

Reference	
Referenced Key	
Replacement Field	
Table Name	diabetes
Matching Field	
Replaced Field	

Window Size Setting

Window Size	10

Table Meta Data - diabetes

S/No.	Column Name	Column Type	Column ...	Nulls All...
1	NRIC	CHAR	9	Yes
2	RACE	CHAR	1	Yes
3	CLIN	CHAR	1	Yes
4	SEX	CHAR	1	Yes
5	DOB	CHAR	10	Yes
6	DOS	CHAR	10	Yes
7	CTYP	CHAR	5	Yes
8	CSRC	CHAR	5	Yes
9	HDUR	CHAR	5	Yes
10	HHYP	CHAR	5	Yes
11	HCATR	CHAR	5	Yes

SQL Query Window

SQL statement

```
select race,dob,dos,sex,clin from diabetes where rownum<=10
```

Results

RACE	DOB	DOS	SEX	CLIN
C	11/21/1948	04/03/1992	F	1
I	08/25/1961	05/30/1992	M	1
C	07/28/1942	04/22/1992	M	1
C	12/12/1927	04/02/1992	F	1
C	11/10/1922	06/29/1992	M	1
C	05/01/1928	05/07/1992	F	1
C	10/08/1932	04/21/1992	F	1
O	01/01/1958	05/14/1992	F	1
C	01/01/1924	04/29/1992	F	1
O	01/01/1958	05/19/1992	F	1

Fig. 5. IntelliClean

5.2 *Experimental Results*

IntelliClean was evaluated using two real world datasets: a Company dataset and a Patient dataset. The Company dataset requires complex matching logic and data manipulation, while the Patient dataset is a much larger dataset and contains a variety of errors.

5.3 *Cleaning the Company dataset*

5.3.1 *Dataset Description*

We first tested our system with a company dataset of 856 records. Each record contains seven fields, namely, Company Code, Company Name, First Address, Second Address, Currency Used, Telephone Number, and Fax Number. Human inspection reveals 60 duplicate records altogether. Problems encountered with this dataset include large number of empty fields, incomplete data, typographical errors, different representations for names and addresses, misuse of data fields and very similar representations for different entities. Table 2 shows two duplicate records which look quite different due to format and representation differences.

Although the number of records is small, cleaning this database proved to be challenging for the following reasons :

- Very complex functions are needed to separate the true and false duplicates which are very similar. This also led to the increase in rules needed to achieve high recall, while

Two inexact duplicate records in Company dataset.

Code	Name	Currency	Telephone	Fax
PT	P. T PRIMA CIRCUIT INDONESIA	SGD	011-708-822323	011-708-812130
(NULL)	PT. PRIMA CIRCUIT INDONESIA	SGD	011-708-822323	(011-708)812130

Address
JL. R.E. MARTADIN,, SEKUPANG-BATAM,INDONESIA 29022
(BATAM FACTORY) Jl. RE. MARTADIN SEKUPANG-BATAM,,BATAM 29022

Table 3
Two very similar records which are not duplicates.

Code	Name	Currency	Telephone	Fax
HIBEXTRA-KL	HIBEXTRA ENGINEERING (M) SDN BHD	MAR	02-08-7034209	02-08-7033866
HIBEXTRA-KL(RM$)	HIBEXTRA TRADING (M) SDN BHD	MAR	02-08-7034209	02-08-7033866

Address
LOT 3, JLN PXS 311/99, BANDAR SUNWAY, 4650 PETALING JAYA SELANGOR DARUL EHSAN, MALAYSIA
LOT 3, JLN PXS 311/99, BANDAR SUNWAY, 4650 PETALING JAYA SELANGOR DARUL EHSAN, MALAYSIA

keeping precision high at the same time. Table 3 shows 2 very similar records which are *not* duplicates. Duplicate identification rules will need to be very complex and precise in order not to classify these records as duplicates.

- A lot of missing vital information. This makes it very difficult to reason about potential duplicates (or even impossible).

5.3.2 Data Cleaning process

The steps taken to clean this dataset are :

(1) Pre-processing.
We first joined the two address fields into a single field as visual inspection reveals that these two fields actually refer to two-halves of the address field and the separation is arbitrary and artificial. The abbreviations used in the Address and Name fields were also standardized using a look-up table. The table contains the commonly used variants and abbreviations of the words found in names and addresses, and their standardized representation.

(2) Processing.
Results were obtained for the first series of tests by running the IntelliClean system with the dataset, varying the number of passes and the number of rules. The metrics generated are the runtime, recall and false-positive error. All the 7 rules used are duplicate identification rules and they are described next.

(3) Human Verification and Validation.
The results of the tests are verified and manually inspected with the log file generated.

5.3.3 Rules and Similarity Measures Used

We used 7 duplicate identification rules for this dataset and several of these rules used a ‘FieldSimilarity’ function adapted from [LLLK99]. This function determines the degree of similarity between the corresponding fields of two records and is defined as follows :

(1) Compute Token Similarity.
  - If two tokens are an exact match, then they have a degree of similarity of 1.
  - Otherwise, if there is a total of x characters in the token, then we deduct $\frac{1}{x}$ from the maximum degree of similarity (DoS) of 1 for each character that is not found in the other token.

(2) Compute Field Similarity.
  - Suppose a field in Record X has tokens $t_{x_1}, t_{x_2}, \ldots, t_{x_n}$, and a corresponding field in Record Y has tokens $t_{y_1}, t_{y_2}, \ldots, t_{y_m}$.
  - Each token t_{x_i}, $1 \le i \le n$, is compared with all the tokens t_{y_j}, $1 \le j \le m$.
  - Let DoS_{x_1}, DoS_{x_2}, $\ldots$, DoS_{x_n}, DoS_{y_1}, DoS_{y_2}, $\ldots$, DoS_{y_m} be the *maximum* of the degree of similarities computed for tokens t_{x_1}, t_{x_2}, $\ldots$, t_{x_n}, t_{y_1}, t_{y_2}, $\ldots$, t_{y_m} respectively.
  - Field similarity for Record X and Y on this field is given by
    $Sim_F(X,Y) = \frac{(\sum_i DoS_{x_i} + \sum_j DoS_{y_j})}{(n+m)}$

Due to space constraints, we show the simplified version of the matching criteria of the 7 duplicate identification rules used :

(1) Rule 1 : Certainty Factor 0.95
  - Matching[1] Currency field
  - and Matching Name field
  - and Matching Telephone field
  - and Matching Fax field

(2) Rule 2 : Certainty Factor 0.85
  - Matching Currency field
  - and FieldSimilarity of the 2 Address fields is at least 0.85
  - and The Code field of 1 record is a sub-string of the other.

(3) Rule 3 : Certainty Factor 0.8
  - Matching Currency field
  - and Matching Code field
  - and FieldSimilarity of the 2 Address fields is at least 0.9
  - and Matching Telephone field after removing all non-numeric characters
  - and Matching Fax field after removing all non-numeric characters

(4) Rule 4 : Certainty Factor 0.8
  - Matching Currency field
  - and Matching Name field
  - and Matching Address field

(5) Rule 5 : Certainty Factor 0.75
  - Matching Currency field
  - and Matching Code field
  - and FieldSimilarity of the 2 Name fields is at least 0.9
  - and More than 75% of the tokens in one record's Address field appears in the other record's Address field.

(6) Rule 6 : Certainty Factor 0.7
  - Matching Currency field
  - and Matching Name field
  - and FieldSimilarity of the 2 Address fields is at least 0.9

(7) Rule 7 : Certainty Factor 0.7
  - Matching Currency field

[1] *Matching* here refers to exact string matches.

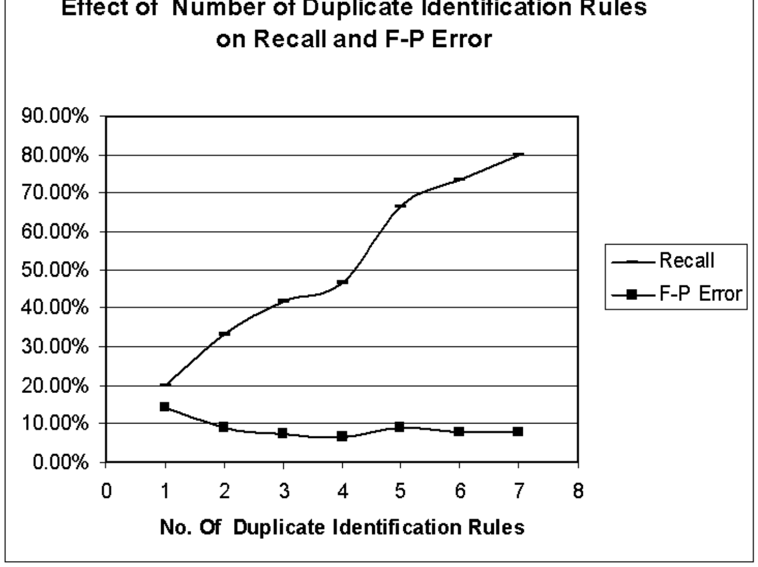


Fig. 6. The Effect of Number of Duplicate Identification Rules on Recall and False-Positive Error

- and Matching Code field
- and Matching Name field
- and FieldSimilarity of the 2 Address fields is at least 0.8

5.3.4 Performance Results

Details of the results are shown in Table 4. Figure 6 shows graphically the effect of the number of rules on recall and false-positive error. The system managed to detect 80% of the duplicates with 7 duplicate identification rules while keeping the false-positive error at less than 8%. Figure 7 shows the effect of the number of rules on runtime. Based on the results, we note that the recall increases with the number of rules, and the more complex rules (which requires more processing time) managed to identify more true duplicates in this case. The runtime of the system depends more on the complexity of the rules rather than the number of rules. Due to the lack of knowledge about the rules of data entry of this dataset, we can only develop 7 duplicate identification rules through visual inspection of sample records. We might postulate that an increase in the number of effective rules can lead to a further rise in recall while maintaining the false-positive error at low levels. This knowledge-based framework thus effectively resolves the recall-precision dilemma.

The window size does not have much effect on recall, unless the window size is comparable to the size of the dataset. This is because the effectiveness of SNM depends on inexact duplicates being close to one another after sorting on a chosen key, and that their proximity depends only on the *first few critical characters* of the key. Consider a simple case of a n-character key string and that two records will never be in the same window during the SNM window scan if any of the first m characters are “far apart”. Assuming that there is a single typographical error in one of the keys, and that the error is equally likely to be at any character of the string, the probability of the character being one of the first few critical characters is m/n. In the case of a 100-character key and the critical characters are the first 3, the probability of two duplicates not being in a same window due to an error in the critical characters is 0.03. Thus, even in the event of having erroneous data in the key, chances are that the records will still appear in a same window during the SNM scanning process. In our example above, the probability that the two duplicate records being detected is 97%. This estimate is arguably

Cleaning Results for Company Dataset using 7 Duplicate Identification Rules and Window Size 10. Each pass uses a different sort key.

Rules Used	Metrics	No. of passes		
		1	2	3
1	Recall(%)	20.00	20.00	20.00
	FP Error (%)	14.29	14.29	14.29
	Time (s)	7.5	12.8	17.6
1,2	Recall(%)	33.33	33.33	33.33
	FP Error (%)	9.09	9.09	9.09
	Time (s)	11.8	17.3	23.7
1,2,3	Recall(%)	36.67	41.67	41.67
	FP Error (%)	8.33	7.41	7.41
	Time (s)	11.9	17.4	23.7
1,2,3,4	Recall(%)	43.33	46.67	46.67
	FP Error (%)	7.14	6.67	6.67
	Time (s)	24.2	31.8	45.8
1,2,3,4,5	Recall(%)	61.67	66.67	66.67
	FP Error (%)	9.76	9.09	9.09
	Time (s)	28.2	39.7	64.9
1,2,3,4,5,6	Recall(%)	65.00	73.33	73.33
	FP Error (%)	9.30	7.55	7.55
	Time (s)	35.9	45.5	68.1
1,2,3,4,5,6,7	Recall(%)	71.67	80.00	80.00
	FP Error (%)	8.51	7.69	7.69
	Time (s)	36.9	48.0	72.1

pessimistic through the use of multiple passes of the SNM with each pass employing a different sort key, which makes the chances of detection even higher.

Table 5 and Figure 8 show the results of our experiments when we vary the window size, the number of passes and the number of rules. Note that recall does not increase much as window size increases from 5 to 30.

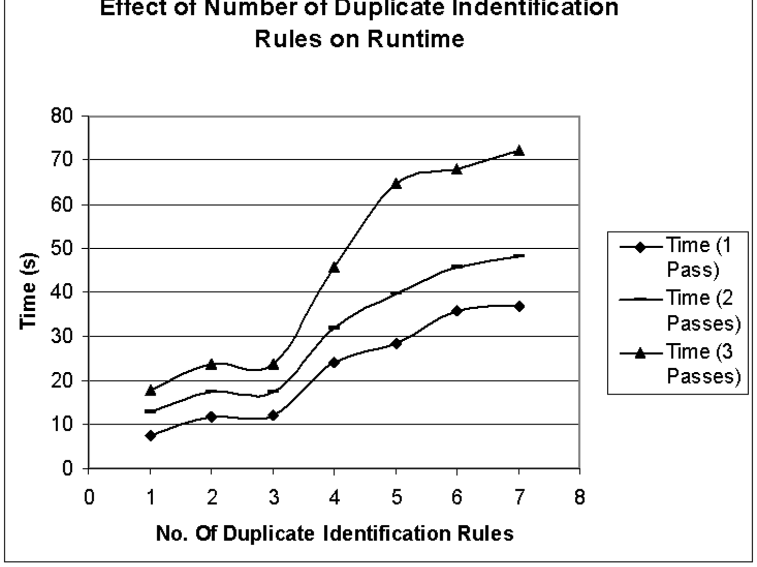


Fig. 7. The Effect of Number of Duplicate Identification Rules on Runtime

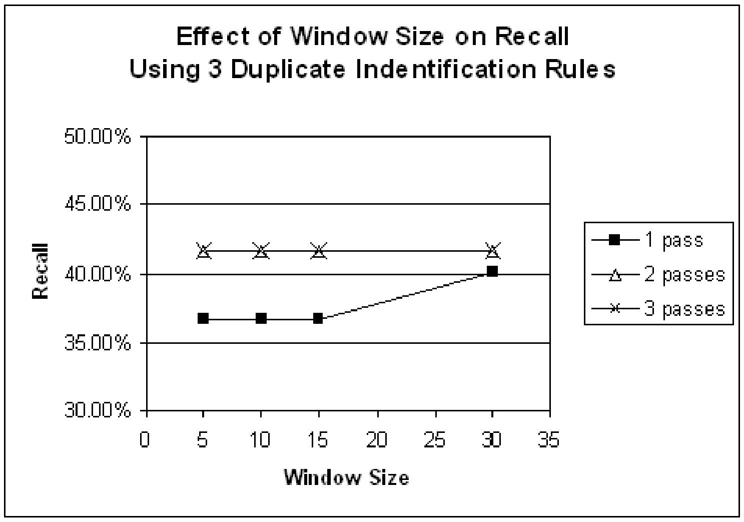


Fig. 8. The Effect of Window Size on Recall

5.4 Cleaning the Patient dataset

5.4.1 Dataset Description

Next, we cleaned a hospital's patient database containing 22122 records. This database contains 60 fields, including the NRIC Number [2], Sex, Date of Birth, Date of Screening, the Clinic Number and various fields containing codes for the results of a diabetes screen test.

[2] The National Registration Identification Card (NRIC) Number is the local equivalent of the Social Security Number of the United States.

Cleaning Results for Company Database With Various Window Sizes

Rules Used	Metrics	Window Size 5			Window Size 10		
		No. of passes			No. of passes		
		1	2	3	1	2	3
1	Recall(%)	20.00	20.00	20.00	20.00	20.00	20.00
	FP Error (%)	14.29	14.29	14.29	14.29	14.29	14.29
	Time (s)	6.5	11.9	15.5	7.5	12.8	17.6
1,2	Recall(%)	33.33	33.33	33.33	33.33	33.33	33.33
	FP Error (%)	9.09	9.09	9.09	9.09	9.09	9.09
	Time (s)	10.0	15.3	21.2	11.8	17.3	23.7
1,2,3	Recall(%)	36.67	41.67	41.67	36.67	41.67	41.67
	FP Error (%)	8.33	7.41	7.41	8.33	7.41	7.41
	Time (s)	10.8	16.2	21.2	11.9	17.4	23.7

Rules Used	Metrics	Window Size 15			Window Size 30		
		No. of passes			No. of passes		
		1	2	3	1	2	3
1	Recall(%)	20.00	20.00	20.00	20.00	20.00	20.00
	FP Error (%)	14.29	14.29	14.29	14.29	14.29	14.29
	Time (s)	7.9	13.4	18.6	8.6	13.6	18.7
1,2	Recall(%)	33.33	33.33	33.33	40.00	40.00	40.00
	FP Error (%)	9.09	9.09	9.09	7.69	7.69	7.69
	Time (s)	12.1	17.6	24.0	14.5	20.4	28.6
1,2,3	Recall(%)	36.67	41.67	41.67	40.00	41.67	41.67
	FP Error (%)	8.33	7.41	7.41	7.69	7.41	7.41
	Time (s)	12.7	18.3	25.9	14.3	20.6	28.8

5.4.2 Data Cleaning process

The steps taken to clean this dataset are :

(1) Pre-processing.
We standardized the date representations of the Date of Birth and Date of Screening fields. Validation checks were also performed on the NRIC Number, Date of Birth, Date of Screening and Sex fields.

(2) Processing.
We defined two duplicate identification rules for this database and one merge/purge

Alerts raised for Patients Dataset

Nature of Alerts	Count
NRIC Check Digit Error	118
NRIC Format Error	10
Sex Field Error	1

rule. The merge/purge rule removes all instances of *exact duplicates* and keeps only one instance. The rules are described after this.

(3) Human Verification and Validation.

5.4.3 Rules Used

The two duplicate identification rules used are :

(1) Duplicate Id. Rule 1 : Certainty Factor 1
- All 60 fields of the record match.

(2) Duplicate Id. Rule 2 : Certainty Factor 0.9
- Matching Date of Screening field.
- and Matching NRIC Number after removing all non-numeric characters.

The merge/purge rule used is defined as :

(1) Merge/Purge Rule
- For all duplicate record groups with certainty factor 1, keep one record from the group and delete the rest.

5.4.4 Performance Results

The alerts raised during the pre-processing stage are shown in Table 6. The pre-processing stage took 118.5 seconds. The results of the processing stage are shown in Table 7. Manual inspection of the database revealed that we achieved 100% recall and precision for this experiment. Although the records identified by Rule 1 of Table 7 form a subset of those identified by Rule 2, we still include this rule as it identifies *exact duplicates*, which we can easily use a Merge/Purge Rule to automatically remove all, except for 1 record. The 8 records (4 pairs) that were exact duplicates were automatically processed in this manner. The rest were marked for manual processing. Generally, we can have more complex merge/purge rules for dealing with more complicated cases. The pre-processing and processing stages (using Rules 1 and 2) took about ten minutes for this dataset.

5.5 Comparative Study

Table 8 compares the methodologies used in IntelliClean with the closest works, namely [LLLK99], [HS95] and [Alv97]. Table 9 gives a comparison of IntelliClean with 2 recent data

Results of Processing the Patients Dataset

Rule Number	Match Criteria	Run Time (s)	No. of Duplicates Found
1	All 60 fields match	370.1	8 (4 pairs)
2	Matching Date of Screening and Numeric String of NRIC	464.7	34 (17 pairs)
1 + 2	(Criteria of Rule 1 OR Rule 2)	503.2	34 (17 pairs)

Table 8
A comparison of IntelliClean with existing data cleaning methodologies.

	LLLK, 1999	HS, 1995	ME, 1997	IntelliClean
Recall-Precision Dilemma	Yes	Probably	Yes	No
Improves with Domain Knowledge	No	Yes	No	Yes
Anomaly Detection	No	No	No	Yes
Anomaly Correction	Maybe	No	No	Yes
Implements	LLLK, 1999	HS, 1995	ME, 1997	LLLK, 1999 HS, 1995 ME, 1997 and more...

Table 9
A comparison of IntelliClean with 2 recently developed data cleaning frameworks.

	VR, 2000	GFSS, 1999	IntelliClean
Anomaly Detection	Through visual inspection	No	By rules in Knowledge-Base
Anomaly Correction	Yes	Maybe	Yes
Knowledge Specification	via GUI	SQL-like syntax	Declarative rules
Optimization Measures	Generating optimized code for transformations	SQL-query optimization techniques	Sliding window & Rete Algorithm

cleaning frameworks described in [RH00] and [HGS99]. Undoubtedly, the power of IntelliClean lies in its ability to analyze, deduce and infer anomalies and duplicates.

6 Conclusion

In this paper, we have presented a generic knowledge-based approach for duplicate elimination in data cleaning. This approach can be applied to any textual databases in any domain and can support a wide variety of cleaning strategies, of which existing methods form a subset. Pre-processing the data scrubs anomalies and performs data standardization. Domain-dependent rules, which are derived from business logic, provides the intelligence during the processing stage. The verification and validation stage entails human inspection of the results using the log generated.

The proposed holistic framework introduces a knowledge representation in the form of rules and the application of the rules via an expert system engine. It allows for effective management and re-use of domain knowledge for data cleaning. The framework also resolves the recall-precision dilemma of existing data cleaning methods through the use of duplicate identification rules that are effective in identifying true duplicates and strict enough to keep out false-positives. This enables data cleaning strategies achieve both high recall and precision. The precision of duplicate elimination strategies is improved by introducing the concept of a certainty factor for a rule. By considering this uncertainty, the number of incorrect merges during the computation of transitive closure can be reduced. Experiments with real-world datasets demonstrated that the knowledge-based approach can achieve high recall and precision efficiently.

Much work still need to be done in this area. Relations spanning several databases give rise to opportunities for anomaly (including functional dependencies, multi-valued dependencies, foreign key constraints violations etc.) and duplicate detection not possible using a single database. We are currently exploring incremental data cleaning techniques and extending the knowledge-based framework for de-duplicating results returned by web search engines.

References

[Alv97] Alvaro E. Monge and Charles P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proceedings of the ACM-SIGMOD Workshop on Research Issues on Knowledge Discovery and Data Mining.* Tucson, AZ, 1997.

[BD83] D. Bitton and D.J. DeWitt. Duplicate record elimination in large data files. *ACM Transactions on Database Systems*, 8(2):255–265, 1983.

[BGF$^+$97] Stéphane Bressan, Cheng Hian Goh, K. Fynn, M. Jakobisiak, Karim Hussein, Henry B. Kon, T. Lee, Stuart E. Madnick, T. Pena, J. Qu, Annie W. Shum, and Michael Siegel. The COntext INterchange Mediator Prototype. In Joan Peckham, editor, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 525–527, Tucson, Arizona, 13–15 June 1997.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *Computing Surveys*, 18(4):323–364, 1986.

[CdGL$^+$99] D. Calvanese, G. de Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A Principled Approach to Data Integration and Reconciliation in Data Warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, June 1999.

[Coh98] William W. Cohen. Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 201–212. ACM Press, 1998.

[FH99] Ernest J. Friedman-Hill. JESS, the Java Expert System Shell, 1999. Available at URL `http://herzberg.ca.sandia.gov/jess/`.

[For82] Charles Forgy. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[GMPQ$^+$97] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[GR98] Joseph Giarratano and Gary Riley. *Expert Systems : Principles and Programming (3rd Edition)*. PWS Publishing Company, Boston, 1998.

[Her96] M. Hernandez. A Generalization of Band Joins and the Merge/Purge Problem. *Technical Report CUCS-005-1995, Department of Computer Science, Columbia University*, February 1996.

[HGS99] Dennis Shasha Helena Galhardas, Daniela Florescu and Eric Simon. An Extensible Framework for Data Cleaning. *INRIA Technical Report*, 1999.

[HP99] Holger Hinrichs and Kirsten Panienski. Experiences with Knowledge-Based Data Cleansing at the Epidemiological Cancer Registry of Lower-Saxony. In *XPS-99: Knowledge-Based Systems. Survey and Future Directions*, Mar 1999.

[HS95] Mauricio A. Hernández and Salvatore J. Stolfo. The Merge/Purge Problem for Large Databases. In Michael J. Carey and Donovan A. Schneider, editor, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 127–138, San Jose, California, 22–25 May 1995.

[HS98] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery, Vol. 2, No. 1*, pages 9–37, 1998.

[KA85] Beth Kilss and Wendy Alvey. Record Linkage Techniques. In *Proceedings of the Workshop on Exact Matching Methodologies, Arlington, Virginia*. Dept of the Treasury, Internal Revenue Service, Statistics of Income Division, May 1985.

[Kim96] Ralph Kimball. Dealing with Dirty Data. *DBMS Online*, September 1996. Available at URL `http://www.dbmsmag.com/9609d14.htm`.

[Lim98] Infoshare Limited. Best Value Guide to Data Standardising. *InfoDB*, July 1998. Available at URL `http://www.infoshare.ltd.uk`.

[LL97] Mong Li Lee and Tok Wang Ling. Resolving Constraint Conflicts in the Integration of Entity-Relationship Schemas. In *Proceedings of the 16th International Conference on Conceptual Modeling, Los Angeles, California, USA*, pages 394–407, Nov 1997.

[LLLK99] Mong Li Lee, Hongjun Lu, Tok Wang Ling, and Yee Teng Ko. Cleansing Data for Mining and Warehousing. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA99)*, pages 751–760, August 1999.

[LSS94] E.-P. Lim, J. Srivastava, and S. Shekhar. Resolving Attribute Incompatibility in Database Integration: An Evidential Reasoning Approach. In Ahmed K. Elmagarmid and Erich Neuhold, editors, *Proceedings of the 10th International Conference on Data Engineering*, pages 154–165, Houston, TX, Feb 1994. IEEE Computer Society Press.

[May99] Arkady Maydanchik. Challenges of Efficient Data Cleansing. *DM Review*, September 1999. Available at URL `http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=1403`.

[ME96] Alvaro E. Monge and Charles P. Elkan. The Field Matching Problem: Algorithms and Applications. In Evangelos Simoudis, Jia Wei Han, and Usama Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, page 267. AAAI Press, 1996.

[MKCWB97] Elisabeth Metais, Zoubida Kedad, Isabelle Comyn-Wattiau, and Mokrane Bouzeghoub. Using linguistic knowledge in view integration: Toward a third generation of tools. *Data & Knowledge Engineering*, 23:59–78, 1997.

[Mos98] Larissa Moss. Data Cleansing: A Dichotomy of Data Warehousing? *DM Review*, February 1998. Available at URL `http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=828`.

[RH00] Vijayshankar Raman and Joseph M. Hellerstein. Potters Wheel: An Interactive Framework for Data Cleaning and Transformation, 2000. `http://control.cs.berkeley.edu/abc/index.html`.

[Ril99] Gary Riley. CLIPS: A Tool for Building Expert Systems, 1999. Available at URL `http://www.ghg.net/clips/CLIPS.html`.

[SSU96] A. Silberschatz, M. Stonebraker, and J. Ullman. Database research: Achievements and opportunities into the 21st century. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(1):52, 1996.

[Wal98] Mary Jo Waller. A Comparison of Two Incremental Merge/Purge Strategies. *Master Thesis, University of Illinois*, 1998.

[Wie93] Gio Wiederhold. Intelligent Integration of Information. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 434–437, Washington, D.C., 26–28 May 1993.

[WM89] Y. Richard Wang and Stuart E. Madnick. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. In *Proceedings of the Fifth International Conference on Data Engineering, February 6-10, 1989, Los Angeles, California, USA*, pages 46–55. IEEE Computer Society, 1989.