Systematic Development of Data Mining-Based Data Quality Tools

Dominik Luebbers[1] Udo Grimmer[2] Matthias Jarke[1,3]

[1]RWTH Aachen, Informatik V (Information Systems), Ahornstr. 55, 52056 Aachen, Germany
{luebbers, jarke}@informatik.rwth-aachen.de
[2]DaimlerChrysler AG, Research & Technology, PO BOX 2360, 89013 Ulm, Germany
udo.grimmer@daimlerchrysler.com
[3]Fraunhofer FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

Data quality problems have been a persistent concern especially for large historically grown databases. If maintained over long periods, interpretation and usage of their schemas often shifts. Therefore, traditional data scrubbing techniques based on existing schema and integrity constraint documentation are hardly applicable. So-called data auditing environments circumvent this problem by using machine learning techniques in order to induce semantically meaningful structures from the actual data, and then classifying outliers that do not fit the induced schema as potential errors.

However, as the quality of the analyzed database is a-priori unknown, the design of data auditing environments requires special methods for the calibration of error measurements based on the induced schema. In this paper, we present a data audit test generator that systematically generates and pollutes artificial benchmark databases for this purpose. The test generator has been implemented as part of a data auditing environment based on the well-known machine learning algorithm C4.5.

Validation in the partial quality audit of a large service-related database at DaimlerChrysler shows the usefulness of the approach as a complement to standard data scrubbing.

Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

1 Introduction

The widespread use of source-integrating, analysis-oriented data warehouses emphasizes the value of quality information in today's business success. Nevertheless a study by the Meta group revealed that 41% of data warehouse projects failed [20]. As one of the main reasons they identified insufficient data quality leading to wrong decisions ("garbage in, garbage out").

It is normally infeasible to guarantee sufficient data quality by manual inspection, especially when data are collected over long periods of time and through multiple generations of database technology. Therefore (semi-)automatic data cleaning methods have to be employed.

Chaudhuri distinguishes domain specific *data scrubbing* tools (that e.g. correct data according to a fixed dictionary of assignments of zip codes to cities or try to link different records that belong to the same real world object [9], [23], [11]) from *data auditing* tools based on data mining techniques [5]. While the first are expensive to develop and maintain the idea of the latter is to adapt data mining algorithms to induce a structure in a large database. By structure we mean any form of regularity that can be found, e.g. a set of rules or decision trees creating a statistically valid decomposition into subclasses. Deviations from this structure can then be hypothesized to be incorrect.

The quality of a data auditing tool heavily depends on the suitability of the employed data mining algorithms for the application domain. It can only be measured when it is testable whether an outlier is a data error or not. Doing this manually requires too much effort for large databases. Therefore, in this paper we present a systematic, domain-driven development method for a data auditing tool. The core of this method is a highly parameterizable artificial test data generator. We exemplify our approach with a test data generator and an auditing tool for a large service-related database at DaimlerChrysler building

on the well-known decision tree inducer C4.5.

This paper is organized as follows:

After a short review of data quality research and the idea of data auditing in sec. 2, sec. 3 gives an overview of our approach and of the example domain. The main contribution of the paper is in sec. 4 which describes a highly flexible test data generator that can be parameterized to simulate data characteristics of the intended application domain. Sec. 5 then demonstrates how the information gained in this phase is systematically exploited to derive a domain-adjusted data auditing tool and suitable measures for error confidence. Finally, sec. 6 reports experiences both with the test data generator and the application of the improved data auditing tool in the automotive domain, sec. 7 gives a brief overview of some related work, and sec. 8 summarizes our conclusions.

2 Data Quality and Data Auditing

As data quality has been recognized as a key success factor for data centric applications, several proposals for data quality management systems (DQMS) have been made. These approaches rely on a definition of the term data quality and on the declaration of measures for different aspects of it. Therefore we first discuss these questions.

2.1 What is Data Quality?

Data quality is a goal-oriented and therefore not formally definable term [17]. The literature speaks of 'fitness for use' [15] or meeting end-user expectations [18]. Data quality is often expressed in a hierarchy of different categories which are then further refined into quality dimensions such as:

- *accuracy* or *correctness*: Does the data reflect the entities' real world-state?
- *completeness*: Does the database contain all relevant entities?
- *consistency*: Are there any contradictory facts in the database?
- *actuality*: Is the data too old for the intended analyses?
- *relevance*: Is the data relevant for the user's information needs?

Additional research has dealt with the question of how data quality can be ensured from a more technical point of view. Hinrichs defines a 5-level architecture that connects commercial tools for data migration with domain-adaptable components [12]. Jarke et al. [16] point out the importance of rich metadata for a successful data quality management system. Wang et al. [25] and English's extensions for a holistic data quality driven approach [7] concentrate on the dynamic aspects of data quality management. Hinrichs defines a 10-phase process model for data warehouse quality assurance based on the ISO 9001:2000 standard [12].

2.2 Data Auditing

We define data auditing as the *application of data mining-algorithms for measuring and (possibly interactive) improving of data quality.* Thereby it closely resembles the term 'Data Quality Mining' introduced by Hipp et al. [14].

Data auditing can be divided into two subtasks, induction of a structural description (*structure induction*), and data checking that marks deviations as possible data errors and generates probable corrections (*deviation detection*).

Both tasks can run asynchronously. This is useful for an application in the data cleansing phase during warehouse loading [2]: While the time-consuming structure induction can be prepared off-line, new data can be checked for deviations and loaded quickly.

Structure induction implicitly assumes that *data quality is sufficiently high to induce a valid entity structure.* By that, data auditing addresses the data quality dimension of consistency.

Furthermore we assume that *deviations from regularities are errors.* Therefore, data auditing is able to approximate a measure for the data quality dimension accuracy. Substituting an erroneously missing value by the suggestion of a data auditing application deals with the completeness dimension as well.

It must be noted that in some cases the above assumptions do not hold: Outliers can be correct and of great importance for analysis. Therefore, the correction of outliers should always be supervised by a *quality engineer* (see section 3.1).

3 Systematic Domain-Driven Development

According to Fayyad, the selection of a suitable data mining algorithm is one of the most difficult tasks in knowledge discovery [8]. This proposition is theoretically supported by the so-called conservation law which states that "positive performance in some learning situations must be balanced by negative performance in others" [24]. An experienced data mining expert with sufficient business domain knowledge is needed to guide the selection and later run the analysis.

To develop an effective data auditing tool it is therefore necessary to find and parameterize data mining-algorithms that are suitable for the specific application domain.

As said before, the usefulness of a data auditing tool is mainly determined by its error detection capability.

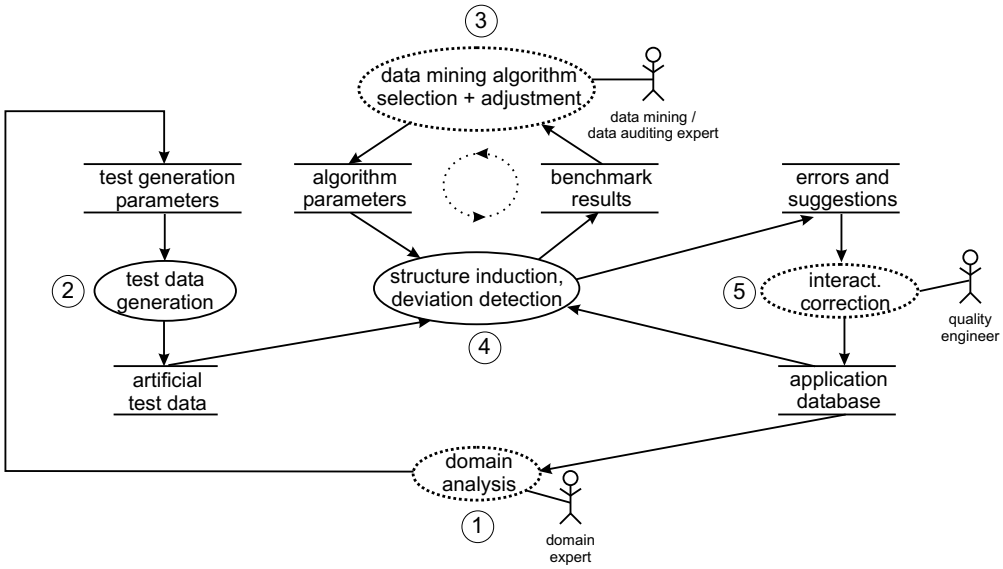

Figure 1: Systematic Domain-Driven Development – Overview

How many of the data entries marked are real errors and how many errors could not be found by the tool (for details about performance measures see section 4.3)? Because of the size of real databases, a manual inspection of the tool's results is far too expensive. The automatic calculation requires the availability of the data source in a clean state that is normally unavailable. Therefore, we ground the selection and adjustment of data mining algorithms in artificially generated test data that mimic structural regularities of the application database.

3.1 Overview of the Approach

The main phases of our approach are shown in figure 1. While the processes (denoted by solid ellipses) are part of the data auditing tool, the dotted ellipses stand for activities by the annotated user roles.

First, characteristic structural properties of the real database must be identified by a domain expert. This leads to parameters that are used to generate artificial test data. Based on this, different data mining-algorithms for structure induction and deviation detection can be tested and, if necessary, adjusted. This process can be iterated until satisfactory benchmark results are obtained. Finally, the customized data auditing tool is used by a *data quality engineer* to identify errors and make suggestions for correct values in the real database. It should be noted that the data quality engineer and the domain expert could be represented by one person.

3.2 Example Domain

Much of this research was done in the context of developing of a data auditing tool for the QUIS (**QU**ality **I**nformation **S**ystem) database at DaimlerChrysler's Global Services and Parts division. QUIS contains both technical and commercial data from the warranty and goodwill periods of all Mercedes-Benz passenger cars and trucks. It is used for different tasks such as product quality monitoring, early error detection and analysis, and reporting. Earlier approaches for applying data mining methods for data quality assessments of the QUIS database have been presented in [10]. QUIS currently holds some 70 GB of data. Regular loads from a variety of heterogeneous source systems need to include data auditing and error correction facilities to ensure QUIS data to be of sufficient quality. Coding errors, misspellings, typing errors, or data load process failures are among the issues that require permanent data auditing and error correction.

Domain experts had defined some characteristic domain dependencies over the QUIS schema which consists of several relations grouped around the central vehicle relation. These dependencies have been used for the validation of our approach. The majority of QUIS attributes are of nominal type, furthermore there are a number of attributes of numerical or date type.

4 Test environment

The test environment justifies selection and adjustment of data mining algorithms. It generates artificial data that simulate structural characteristics of the application database, pollutes this data in a controlled and logged procedure, runs the data auditing tool and evaluates its performance by comparing the deviations of the dirty from the clean database with the detected errors. The overall structure of this test environment is shown in figure 2.

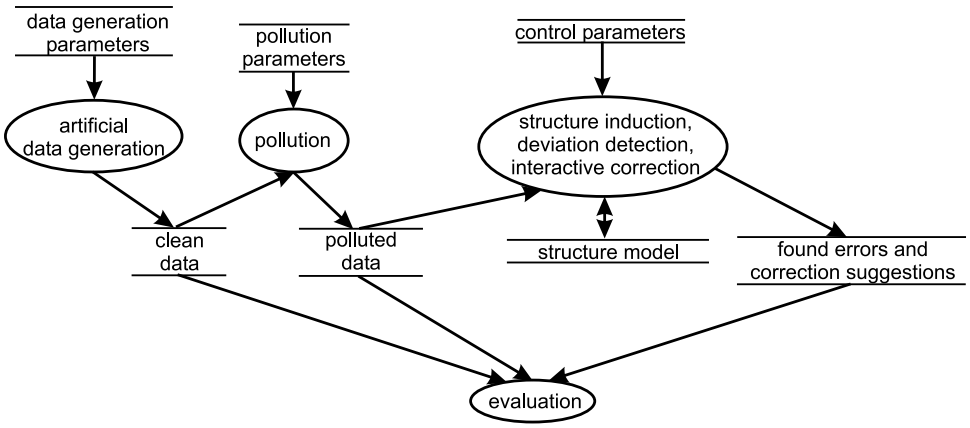

Figure 2: Overall structure of the test environment

4.1 Generation of Artificial Test Data

Motivated by the expert-identified dependencies in the QUIS database we decided to develop a rule-pattern-based test data generator that is expressive enough to simulate QUIS characteristics and sufficiently flexible to be used in other domains as well.

After defining a schema for the target relation with domain ranges for each attribute, the test data generator creates instances of rule patterns randomly according to some user-defined parameters and generates data that matches these rules.

4.1.1 Generation of Rules

We define the structure of rule patterns inductively as consisting of test data generator formulae (TDG-formulae) built of atomic TDG-formulae.

Definition 1 (atomic TDG-formulae) *Let A and B be numerical or nominal attributes and let a_1 be a numerical or nominal domain value. Furthermore let N and M be numerical attributes and let n be a numerical domain value. Then*

- $A = a_1$, $A \neq a_1$, $N < n$, $N > n$, A `isnull`, A `isnotnull`, *(propositional)*
- $A = B$, $A \neq B$, $N < M$ *and* $N > M$ *(relational)*

are called atomic TDG-formulae.

Further we define TDG-formulae as finite conjunctions or disjunctions of atomic formulae:

Definition 2 (TDG-formulae)

- *Each atomic TDG-formulae is a* TDG-formulae.
- *Let $n \in \mathbb{N}$ and $\alpha_1, \ldots, \alpha_n$ be* TDG-formulae. *Then $\alpha_1 \vee \ldots \vee \alpha_n$ and $\alpha_1 \wedge \ldots \wedge \alpha_n$ are* TDG-formulae.

Finally, a TDG-rule is an implication between two TDG-formulae:

Definition 3 (TDG-rule) *Let α and β be TDG-formulae. Then $\alpha \rightarrow \beta$ is a* TDG-rule.

4.1.2 Generation of a Natural Rule Set

Unfortunately, rules constructed according to these definitions do not necessarily comply with a human-generated set of meaningful rules. Consider the following TDG-rules:

$$\begin{aligned} A = Val1 &\rightarrow A = Val2 \\ A = Val1 \wedge A = Val2 &\rightarrow B = Val1 \\ A = Val1 &\rightarrow A \neq Val2 \end{aligned}$$

While the first two rules are contradictory, the last one is tautological.

The creation of such rules should be avoided if the number of generated rules is intended to reflect the structural strength of the data. Therefore we extend the definitions of TDG-formulae and -rules by semantic restrictions.

Definition 4 (Natural TDG-formula) *Let α be a TDG-formula. α is a* natural TDG-formula *iff one of the following holds:*

- *α is an atomic TDG-formula and α is (constrained by the domains of the target schema) satisfiable.*
- *$\alpha = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ and the following holds:*
  - *$\forall i : \alpha_i$ is a natural TDG-formula,*
  - *α is satisfiable and*
  - $\forall i : \alpha_i \not\Leftarrow \bigwedge_{\substack{j \in \{1,\ldots,n\} \\ i \neq j}} \alpha_j$
- *$\alpha = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$ and the following holds:*
  - *$\forall i : \alpha_i$ is a natural TDG-formula and*
  - $\forall i : \alpha_i \not\Leftarrow \bigvee_{\substack{j \in \{1,\ldots,n\} \\ i \neq j}} \alpha_j$

This ensures that a subformula of a disjunction or conjunction is not already implied by the other subformulae. For a conjunction we furthermore demand

satisfiability. This implies inductively satisfiability of the disjunctions as well. Similarly for TDG rules:

Definition 5 (Natural TDG-rule)
A TDG-rule $\alpha \rightarrow \beta$ is called a natural TDG-rule *iff*

- *α and β are natural TDG-formulae,*
- *$\alpha \wedge \beta$ is satisfiable and*
- *$\alpha \not\Rightarrow \beta$*

Nevertheless it is possible to construct a set of rules that are mutually contradictory:

$$\begin{aligned} A = Val1 &\rightarrow B = Val1 \\ A = Val1 &\rightarrow B = Val2 \end{aligned}$$

Additionally it is easy to formulate a rule that is redundant in the presence of a second one:

$$\begin{aligned} A = Val1 \wedge B = Val2 &\rightarrow C = Val1 \\ A = Val1 &\rightarrow C = Val1 \end{aligned}$$

To avoid the construction of these rule sets one could formulate the restriction, that to a given rule set $\mathcal{R}$ the rule $R = \alpha \rightarrow \beta$ should be added only if R is not logically implied by $\mathcal{R}$ ($\mathcal{R} \not\models R$) and $\mathcal{R} \cup \{\alpha\}$ is satisfiable.

As we will see in the next section it is expensive to check this condition. Therefore the definition of a *natural rule set* only employs a pairwise check of rules for contradiction and tautology.

Definition 6 (Natural rule set)
Let $\mathcal{R} = \{\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n\}$ be a set of natural TDG-rules $\alpha_i \rightarrow \beta_i$.
$\mathcal{R}$ is called a natural rule set *iff for two different rules $\alpha_i \rightarrow \beta_i$ and $\alpha_j \rightarrow \beta_j$ with $\alpha_j \Rightarrow \alpha_i$ the following holds:*

- *$\alpha_j \wedge \beta_i \wedge \beta_j$ is satisfiable and*
- *$(\alpha_j \wedge \beta_i) \not\Rightarrow \beta_j$.*

If the premise of a rule implies the premise of another rule, the consequences must not be contradictory. Because of $\alpha_j \Rightarrow \alpha_i$ the satisfiability of $\alpha_j \wedge \beta_i \wedge \beta_j$ implies already the satisfiability of $\alpha_i \wedge \alpha_j \wedge \beta_i \wedge \beta_j$. Furthermore a rule $\alpha_j \rightarrow \beta_j$ introduces a new dependency only if $\alpha_j \wedge \alpha_i \wedge \beta_i$ (which is by assumption equivalent to $\alpha_j \wedge \beta_i$) does not imply the consequence β_j.

These constraints leave a wide space for possible rule sets. Therefore, the rule generation process can be further parameterized to govern the complexity of a rule (e.g. nesting depth or number of atomic subformulae).

α	$\widetilde{\alpha}$
$A = a$	$A \neq a \vee A$ `isnull`
$A \neq a$	$A = a \vee A$ `isnull`
$A < a$	$A > a \vee A = a \vee A$ `isnull`
$A > a$	$A < a \vee A = a \vee A$ `isnull`
A `isnull`	A `isnotnull`
A `isnotnull`	A `isnull`
$A = B$	$A \neq B \vee A$ `isnull` $\vee B$ `isnull`
$A \neq B$	$A = B \vee A$ `isnull` $\vee B$ `isnull`
$A < B$	$A > B \vee A = B \vee$ A `isnull` $\vee B$ `isnull`
$A > B$	$A < B \vee A = B \vee$ A `isnull` $\vee B$ `isnull`
$\alpha_1 \wedge \ldots \wedge \alpha_n$	$\widetilde{\alpha_1} \vee \ldots \vee \widetilde{\alpha_n}$
$\alpha_1 \vee \ldots \vee \alpha_n$	$\widetilde{\alpha_1} \wedge \ldots \wedge \widetilde{\alpha_n}$

Table 1: TDG-negation of a TDG-formula α

4.1.3 A Pragmatic Satisfiability Test for TDG-formulae

In ordinary propositional logic the validity of the sentence $\alpha \Rightarrow \beta$ is equivalent to the unsatisfiability of $\alpha \wedge \neg\beta$. As we did not include negation in the definition of a TDG-formula, and due to the proper handling of null values, we can instead associate a TDG-formula $\widetilde{\alpha}$ to a TDG-formula α, so that α is true iff $\widetilde{\alpha}$ is false. Table 1 shows this *TDG-negation* which reduces the validity of $\alpha \rightarrow \beta$ to the unsatisfiability of $\alpha \wedge \widetilde{\beta}$.

Due to space constraints only the main ideas of the satisfiability test are described here. First, the TDG-formula α is transformed into disjunctive normal form. α is satisfiable iff one of these disjuncts is satisfiable. We thereby reduce the problem to a satisfiability test for a conjunction of atomic TDG-formulae.

The main idea of the procedure is to initialize the *current domain ranges* of every attribute defined in the schema for the target table with their domain ranges and then successively restrict them by integrating the constraints of each atomic TDG-formula in the conjunction. This is straightforward for propositional formulae. The integration of relational constraints expressed by a relational formula are reflected by the instantiation of links between attributes while considering the transitive nature of the operators $<$, $>$ and $=$. According to these links domain restrictions are propagated to all dependent attributes.

It can be proven that the unsatisfiability test of TDG-formulae is correct but there are rare (and in practice irrelevant) cases in which the algorithm believes a formula to be satisfiable although there is no value assignment to attributes under which it evaluates to true.

4.1.4 Data Generation According to Rules

Given a schema for the target table and a rule set, a number of records has to be created that follow this rule set. This is done by selecting values for each at-

tribute according to independent probability distributions and successively adjusting these guesses by rules that are violated. Our system offers uniform, normal and exponential distributions that can be parameterized by the user.

First experiments showed that an independent sampling of the initial values does not lead to a satisfactory model of the QUIS database. Hence, we developed a method for the intuitive specification of multivariate start distributions based on the graphical representation of stochastic dependencies among attributes in Bayesian networks.

4.2 Controlled Data Corruption

Components in the test environment, each parameterized with an activation probability, simulate the strategies for identification and analysis of different forms of data pollution as defined by Dasu [6] and Hernandez [11] :

- *Wrong value polluter*: Assigns a new value to an attribute according to a probability distribution defined in the same way as in section 4.1.4.
- *Null-value polluter*: Replaces the value of an attribute by a null value.
- *Limiter*: Cuts off a numerical value according to a maximal or minimal bound.
- *Switcher*: Switches the values of two attributes.
- *Duplicator*: Duplicates (or deletes) a record.

4.3 Performance Parameters

Error detection by a data auditing tool can be summarized by a 2x2 matrix:

	tool's opinion	
	incorrect	correct
incorrect data	true positive	false negative
correct data	false positive	true negative

The vertical dimension reflects the number of errors generated by controlled data corruption. The horizontal dimension reflects the number of errors found by the data auditing tool. Ideally, the figures of these dimensions fall together.

To quantify data audit quality we use the two measures *specificity* and *sensitivity*. The specificity quantifies how many of the error free records have been marked as such, the sensitivity expresses the ratio of the truly found errors by the number of records that have been corrupted.

The information retrieval literature often uses the measures precision and recall. While precision is a synonym for specificity, we favor sensitivity over recall as it is independent from the prevalence (the total ratio of errors in the table).

A perfect data auditing tool has values of 1 for sensitivity and specificity, in practical environments only lower values can be reached. The importance of a high value for a measure depends on the intended use of the tool: If it is used as a data screening tool that marks deviations to be controlled manually later a high sensitivity is important. If it is necessary to integrate new data very quickly in a data warehouse and filter only records that are incorrect with a high probability, a high value for specificity is recommended.

Besides its capability to mark suspicious records, our data auditing tool can be used to propose a correction. The influence of following this correction on the quality of the data set can be summarized in the following 2x2 matrix:

		after correction	
		correct	incorrect
before correction	correct	a	b
	incorrect	c	d

We employ a very simple formula to measure the improvement (or degradation) achieved by replacing an identified error by the proposed value: It consists of the difference between the number of errors before and after the correction normalized by the number of errors before correction: $\frac{(c+d)-(b+d)}{c+d}$

5 Data Auditing Tool

The quality of a data auditing tool heavily depends on the suitability of the employed data mining algorithm for the application domain. The test environment now serves to evaluate the suitability of different data mining algorithms for a given domain. Here, we restrict ourselves to data auditing tools based on the following strategy:

For each attribute in the relation to be audited, a classifier is induced that describes the dependency of this *class attribute* from the other attributes (called *base attributes* in this context). A record can be checked for deviations by comparing its observed class value with the *predicted value* for each classifier.

It is easy to customize this general procedure by domain knowledge: If it is known that an attribute does not influence the value of a class attribute, it can be removed from the set of base attributes for the classifier.

A dependency model for a nominal class attribute is usually called a classifier while it is called a regression model if the class attribute is numerical. We thus call our method the *multiple classification / regression approach*.

Inside this framework, it is possible to choose different algorithms to induce dependency models between the base and class attributes.

For the QUIS domain we evaluated different alternatives (instance based classifiers, naive Bayes classifiers, classification rule inducers, and decision trees).

This led to the decision to base our structure inducer and deviation detector on the well-known decision tree package C4.5 [22].

To allow for the induction of decision trees for numerical class attributes, these attributes are discretized into equal frequency bins before the induction process.

In the following, we briefly describe the main algorithms for decision tree induction and classification according to C4.5. Then, a formula for the quantification of the tool's belief in the faultiness of a record is developed. In a third subsection we discuss some adjustments of the data mining procedures required for data auditing.

5.1 Decision Tree Induction and Classification according to C4.5

5.1.1 The Basic Algorithm

Decision tree induction according to ID3 [21] can be seen as precursor of the C4.5 algorithm.

Let $T = \{t_1, \ldots, t_n\}$ be a set of n training instances each with values for the base attributes $A_1, \ldots, A_m$ and the class attribute C. For simplicity, we assume at first that the domains of all of these attributes are nominal. W.l.o.g. they can be denoted by $\mathsf{dom}(A_i) = \{1, \ldots, n_i\}$. By T_{pred} we denote the set of all instances of T for that *pred* evaluates to true.

The main idea is to successively partition T into subsets with a more homogenous class distribution. This is done by a simple recursive top-down-algorithm.

We start with a root that is labelled with T. We then select an attribute A_i that best separates T into the partitions $T_{A_i=i}, \ldots, T_{A_i=n_i}$. For each of these partitions, new nodes that become children of the root node are created and labelled. The separation process is called recursively until all instances of a partition have the same class or there are no more attributes left to further separate the set.

To narrow the search space of possible decision tree classifiers, ID3 uses a greedy approach to select the split attribute A_i. This selection is based on the expected loss of entropy, called *information gain*, which can be defined for set of instances S and a base attribute A_i as follows:

$$\mathsf{info\text{-}gain}(S, A) := \mathsf{entr}(S) - \sum_{j=1}^{n_i} \frac{|S_{A_i=j}|}{|S|} \cdot \mathsf{entr}(S_{A_i=j})$$

The entropy of an instance set S is defined as:

$$\mathsf{entr}(S) := -\sum_{j=1}^{n_C} \frac{|S_{C=j}|}{|S|} \cdot \log_2(\frac{|S_{C=j}|}{|S|})$$

The classification of an unseen instance by a decision tree is straight-forward: Starting at the root node a path to a leaf is selected according to the values of the base attributes. The majority class of the instances, that the reached leaf is labelled with, becomes the predicted class value.

5.1.2 Improvements by C4.5

The ID3 information gain measure systematically favors attributes with many values over those with fewer values [22]. C4.5 divides the information gain by *split information*, which leads to a measure called *information gain ratio*. The split information of an instance set S and an attribute A_i is defined as follows:

$$\mathsf{split\text{-}info}(S, A_i) := -\sum_{j=1}^{n_i} \frac{|S_{A_i=j}|}{|S|} \cdot \log_2(\frac{|S_{A_i=j}|}{|S|})$$

Numerical base attributes are handled in C4.5 by introducing binary auxiliary variables that represent split points taken from the set of all occurring values for the numerical attributes in the schema.

In contrast to many other machine learning algorithms C4.5 is able to handle training instances with missing values, by replacing the observed attribute value of an instance by its expected value calculated from the instances without null value. This approach requires the possibility to 'distribute' a training instance over several branches of an inner node in the decision tree.

In its pure form, a decision tree classifier is exposed to the danger of overfitting. Hence, C4.5 postprocesses the generated decision tree in a *pruning* step. The widely used *subtree replacement* replaces a subtree by a single leaf if this reduces the classification error (i.e. the ratio of misclassified instances). Calculating this measure on the basis of the training instances is usually too optimistic. The prediction of a class value in a leaf of the decision tree is based on the set of training instances it is labelled with. These instances can be seen as a sample from a population and the equality resp. inequality of a class value with the predicted class can be interpreted as the outcome of a Bernoulli experiment.

The following definition of a *pessimistic classification error* shows how the size of the sample is taken into account:

- If k is a leaf and c is the majority class of S:

$$\mathsf{pessError}(k) := \mathsf{rightBound}(1 - \frac{|S_{C=c}|}{|S|}, |S|)$$

- If k is an inner node and $k_1, \ldots, k_l$ are its sons with the instance set labels $S_1, \ldots, S_l$:

$$\mathsf{pessError}(k) := \sum_{j=1}^{l} \frac{|S_j|}{|S|} \cdot \mathsf{pessError}(k_j)$$

Here, $\mathsf{rightBound}(p, n)$ denotes the right bound of the confidence interval for the true probability of occurrence given the observed probability p and a sample size of n. The confidence level of this interval can be parameterized.

5.2 Error Confidence in Data Audits

In data audits, each induced decision tree is used to compare the observed value of an instance with the value predicted by the classifier. If these values do not match the record is marked as a possible data error.

The prediction of a decision tree is based on one or more (in the presence of null values) leaves. Based on the class distributions of the instance sets these leaves are labelled with, one can easily extend the prediction of a decision tree to the calculation of a class distribution. This allows to quantify the degree by which an observed value differs from the expectations and hence the strength of the data auditing tool's belief that an error has been found.

For the following explanations let P be the class distribution, c the observed class, and $\hat{c}$ the predicted class (which means that $P(\hat{c}) \geq P(c')$ for all $c' \neq \hat{c}$.

At first glance it seems natural to take $1 - P(c)$ as a measure for the deviation. But this would assign equal scores to the probability distributions $P_1 = (0.2, 0.2, 0.2, 0.1, 0.3)$ and $P_2 = (0.2, 0.8, 0.0, 0.0, 0.0)$ assuming the first class value has been observed, although an error is more apparent in the second case.

A second idea is to simply take the probability of the predicted class $P(\hat{c})$, if $c \neq \hat{c}$, as an error measure. Again, this can be criticized with the distributions $P_1 = (0.0, 0.1, 0.9)$ and $P_2 = (0.1, 0.0, 0.9)$ that should not lead to equal error scores when the first class value is observed.

The last example motivates the idea of utilizing the difference $P(\hat{c}) - P(c)$.

As in the case of pessimistic classification error, the number of training instance the calculation of P is based on should be taken into account in addition to the relative frequencies expressed in the probability distribution P. This leads to the following definition of the *error confidence* of a record *wrt. one classifier*:

Definition 7 (Error confidence wrt. one classifier)
Let P be the predicted class distribution, n the number of instances this prediction is based on, and c and $\hat{c}$ as used above.
Then, the error confidence wrt. one classifier *can be calculated as:*

$$\mathsf{errorConf}(P, c) := \max(0, \quad \mathsf{leftBound}(P(\hat{c}), n) - \mathsf{rightBound}(P(c), n))$$

where $\mathsf{leftBound}(p, n)$ and $\mathsf{rightBound}(p, n)$ denote the left resp. right bounds of the confidence interval for the true occurrence probability given the observed probability p and the sample size n.

The error confidence values for each classifier have to be combined to give a measure for the overall error confidence for a record. To score a deviation, Hipp adds the precision values of all violated association rules [14]. This addition is, strictly speaking, only valid if all rules predict values for the same attributes. Otherwise, the values that are prescribed by one rule might inhibit the applicability of another rule.

Hence, we simply choose the maximum of all error confidences wrt. classifiers as an overall error confidence for the record.

Definition 8 (Error confidence for a record)
Let $\mathsf{errorConf}_1, \ldots, \mathsf{errorConf}_k$ be the calculated error confidences wrt. different classifiers.
The overall error confidence for a record is defined as:

$$\mathsf{errorConf} := \max_i \{\mathsf{errorConf}_i\}$$

It should be emphasized that the error confidence measure can be used with each classifier that both outputs a predicted class distribution and the number of training instances this prediction is based on. This independence from C4.5 makes it usable in data auditing tools for domains that require different data mining algorithms.

5.3 Calculation of a Proposed Correction

In addition to its use for detecting data errors, the predicted values generated by the classifiers induced can be employed directly as suggested corrections. To calculate the quality of correction measure defined in section 4.3, we replace a suspicious value according to the prediction of the classifier with the highest error confidence.

In interactive error correction, the predicted distributions of all classifiers that indicate a data error can be useful in finding the true reason for a possible error. This is because a difference between an observed and predicted value sometimes lays in erroneous base attribute values.

5.4 Adjustments for the Data Auditing Context

First experiments with data from the test data generator revealed that the unadjusted use of the decision tree inducer was problematic for the following reasons:

- If the class attribute is only roughly dependent on the base attributes, a highly complex decision tree is generated and pruned afterwards to a single node tree. When the number of base attributes and the size of their domains increases, the separate construction of a perfect decision tree is infeasible. Therefore, the induction and pruning phase have to be combined.

- Even very high values for the confidence level used in the calculation of the pessimistic classification error sometimes do not lead to a subtree replacement although the partitioning into several subsets does not increase the error detection capability. Hence, a different pruning measure has to be developed.

Low error confidence values are mostly not useful in reality. If we let the user restrict his interest by giving a minimal confidence for detected errors, the system can easily calculate the minimal number minInst of instances of one class that have to occur in a leaf of the decision tree. This number can be used in a prepruning strategy to prevent a training instance set from being further partitioned when there is not at least one subset with minInst instances of one class.

The quality of a classifier is determined by its classification error on unseen test data. This criterion is approximated in C4.5 by the pessimistic classification error on training data. In data auditing, the prediction of a class value is only the first step that is followed by the calculation of an error confidence. Therefore, it is reasonable to base the pruning criterion on this measure by defining an *expected error confidence*.

Definition 9 (Expected error confidence) *Let k be a leaf in a decision tree, S the associated set of training instances, and P the class distribution at k. Then, the expected error confidence is defined as:*

$$expErrorConf(k) := \sum_{c=1}^{n_C} \frac{|S_{C=c}|}{|S|} \cdot errorConf(P, c)$$

If k is an inner node with sons $k_1, \ldots, k_l$ that are labelled with $S_1, \ldots, S_l$, the expected error confidence of k can be calculated as:

$$expErrorConf(k) := \sum_{j=1}^{l} \frac{|S_j|}{|S|} \cdot expErrorConf(k_j)$$

To avoid the construction of a space-consuming unpruned decision tree, the expected error confidence is used in an integrated pruning strategy that replaces a subtree by a single leaf whenever this transformation leads to a higher value for expErrorConf.

It is straightforward to represent an induced decision tree as a set of rules from the root to its leaves. If the dependency of a class attribute on its base attributes is very punctiform, it is often useful to reduce this set to the rules that do not have an expected error confidence of zero and thereby cannot contribute to an error detection.

The rule sets generated by all classifiers in the multiple classification / regression approach build the *structure model* of the data. In database terminology it can be seen as a set of integrity constraints that must hold with a given probability.

To summarize, we again list the above discussed adjustments of the C4.5 algorithm for its use as a structure inducer in the context of the QUIS database:

- We avoid the construction of insignificant subtrees during decision tree induction based on a calculated minimal number of instances of one class (minInst) that has to be contained in one partition.
- We replace the C4.5 pruning criterion, that consists mainly of a pessimistic guess for the misclassification rate, by the so-called expected error confidence that takes into account the actual use of the classifier in the data auditing context. Furthermore we employ this pruning criterion already during the tree construction phase. This integration extends the pre-pruning strategy mentioned above.
- We transform the decision tree into an equivalent rule set and delete all rules that are not useful for error detection.

6 Evaluation

To show the applicability of our data auditing tool in the QUIS domain, we present some evaluation results.

Besides its use as a benchmarker during development, the test environment allows for a thorough exploration of the application boundaries for the data auditing tool. Because of its large parameter space one can analyze the influence of certain structural aspects in a database on the tool's effectiveness. This allows for the definition of conditions under which the application of the developed data auditing tool can be utilized profitably. In this sense our approach resembles Aha's proposals for the generalization of case studies for classifiers [1].

First, we show some evaluation results collected by the test environment for different generator and polluter parameter settings. Then we report on some experiences gained with applying the tool to a sample of the QUIS database.

6.1 Evaluation on Different Test Data Settings

For the following we fix a minimal error confidence of 80%. This leads to high values for specificity of about 99% in all parameter settings described. We therefore concentrate on the influence of these settings to sensitivity. Furthermore, it was observed that the quality of correction is highly correlated to sensitivity.

We start with a basic parameter configuration that prescribes 6 nominal attributes with different domain sizes, 1 date type and 1 numeric attribute. Furthermore, we specify one multivariate nominal and 5 univariate start distributions of different kinds. We use the test data generator to create 10000 records based

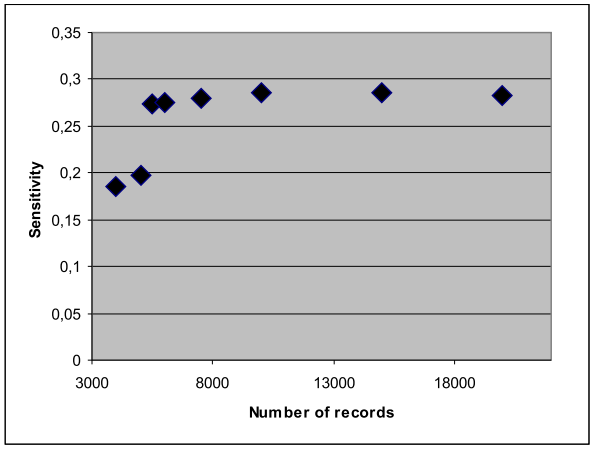

Figure 3: Influence of number of records on sensitivity

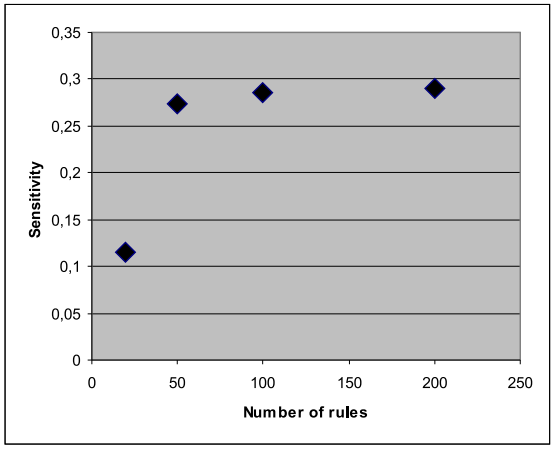

Figure 4: Influence of number of rules on sensitivity

on 100 randomly generated rules and apply a variety of pollution procedures with different activation probabilities.

First, we vary the number of records in the table. The influence of this parameter on sensitivity is shown in figure 3. It shows a rising tendency of the sensitivity with a growing number of records up to a value of nearly 0.3, which means that almost 30% of the true errors could be identified. The more sample records are exposed to the structure induction procedure the more definitely regularities can be formulated. This is numerically reflected by the influence of the sample size to the calculation of the error confidence (see section 5.2).

A sensitivity of 30% may looks disappointing on the first glance, but one should keep in mind that data auditing tools can principally only detect errors that are deviations from regularities, which is not the case for all error types. Our data pollution procedure corrupts data values regardless of the fact whether they are part of a generated rule or not. Therefore it is obvious that

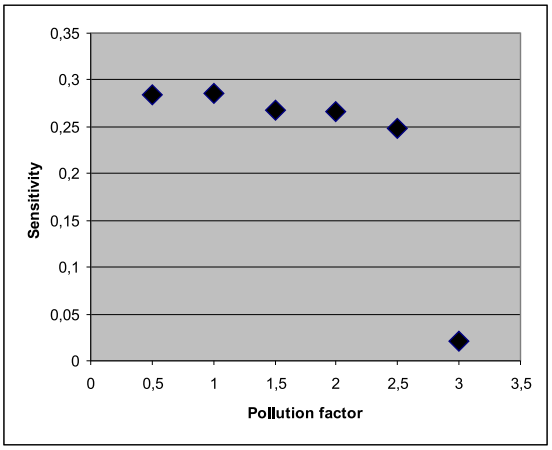

Figure 5: Influence of pollution factor on sensitivity

only a fraction of these errors can be found by our data auditing tool. This makes clear that data auditing tools should always be accompanied by domain specific data scrubbing tools to form a holistic data quality management.

The jump in the trend of the curve at 6000 records can be explained as an effect of the minimal error confidence: A training set of less than 6000 records does not induce rules that fall beyond the minimal error confidence limit. As these rule are deleted, they cannot be used for error detection.

Figure 4 shows the influence of the number of rules (which can be seen as a measure for the structure strength due to the restriction to natural rule sets) on sensitivity. As expected, the diagram reveals that the more constraints are imposed on the data the easier it is to identify errors based on deviation detection. Nevertheless, it shows that even for highly regular data sets a sensitivity value of 0.3 is not exceeded. This can be explained by the fact that the rules constructed from the hierarchical structure of a decision tree are not able to fully express all dependencies that can be formulated by a set of TDG-rules.

As a last example we vary the activation probabilities of the employed pollution procedures (see section 4.2) by multiplying them with a common *pollution factor*. Its influence on the tool's sensitivity is shown in figure 5. It is obvious that the more corrupted the table is, the less valid rules that lead to correct error identifications can be induced. Again, the drop in the trend at pollution factor 3 can be explained as a consequence of the minimal error confidence limit. If the data is too erroneous to find partitions with sufficiently homogenous class distributions, no useable rules for deviation detection can be formulated.

6.2 Auditing of a QUIS Sample

To evaluate the applicability of the data auditing tool on real-world data, we chose a table of the QUIS

database that describes the composition of all industry engines manufactured by Mercedes-Benz. It contains 8 attributes and about 200000 records. The attributes code the model category of each individual engine and its production date.

Running the error detection process lasted about 21 minutes on an Athlon 900Mhz system and revealed about 6000 suspicious records. These records were ranked according to their associated error confidence and cross-checked by domain experts selectively.

For example, the following dependency between the two attributes BRV and GBM was induced:

$$\mathsf{BRV} = 404 \rightarrow \mathsf{GBM} = 901$$

It was based on 16118 instances. One instance, however, contradicts the rule: It has got a value of 911 for the GBM attribute. The data auditing tool assigns an error confidence of 99,95% to this instance and ranks it first in the sorted list of suspicious records.

Similarly, the rule

$$\mathsf{KBM} = 01 \wedge \mathsf{GBM} = 901 \rightarrow \mathsf{BRV} = 501$$

that relies on 9530 records, results in a lower confidence measure of 92% for a deviating instance.

An exact quantification of real-world sensitivity and specificity by domain experts turned out to be too expensive even for this relatively small excerpt. Nevertheless, experts agreed that the identification of the deviations with the highest error confidences is a highly valuable information for data quality engineers.

7 Related Work

The application of data mining algorithms for data cleaning has been investigated by a number of researchers.

Hipp et al. [14] use scalable algorithms for association rule induction and define a scoring that rates deviations from these rules based on the confidence of the violated rules. Unfortunately, association rules cannot directly model dependencies between numerical attributes.

Hinrichs et al. describe a framework that allows for the use of different algorithms from the data mining library $\mathcal{MLC}$++ and can be easily integrated and customized by defined meta data structures [13]. As their data mining algorithms are not adjusted for data auditing, performance is only sufficient for very small data volumes.

Brodley and Friedl employ data mining during data preprocessing in the KDD process [4]. By filtering possibly erroneous training data before inducing a classifier, they are able to significantly decrease the misclassification rate. However, their goal is to improve the resulting classifier, not to detect errors in the training data. Given that the training data set is sufficiently large it is therefore reasonable to remove all suspicious instances while their denomination as an error is not justified.

Marcus and Maletic apply different data mining algorithms for automatic error detection in a personnel database of the Navy Research and Development Center [19]. Unfortunately, the proposed use of so-called ordinal association rules is only suitable for databases with many attributes of decimal or date type.

Much literature deals with definitions and detection algorithms for data outliers that can be seen as exceptions (e.g. [3]). However, these approaches usually require the definition of a distance function between two data items, which is not an easy task for databases with mainly nominal attributes.

8 Conclusions

The lessons from this work can be summarized as follows: Successful data auditing requires sophisticated error detection facilities in combination with high-quality proposals for replacing erroneous data values. As the full database is to be screened, only data mining algorithms that scale well with the size of training sets can be employed. For both performance and quality, the search space of such inducers should be pre-restricted by a domain analysis which also helps to construct the test environment. Finally, a data auditing tool should work both when training sets and test data are separate and when there is only a single database which serves both for training and data audit.

These demands are covered by our approach for the systematic development of a data auditing tool that can be iteratively adjusted by the use of generated artificial test data. This rule-based test data generator is highly parameterizable and allows the simulation of very different kinds of structures that can be found in real-world data.

We demonstrated the approach for the domain of a large service-related database at DaimlerChrysler and adjusted the well-known C4.5 algorithm for being used in this context.

Experiments with artificial test data revealed a very high specificity and a sensitivity of up to 30% of the data auditing tool. Keeping in mind that data auditing tools can find only the fraction of data errors that are deviations from regularities this is a promising value. Nevertheless these results make clear, that further mechanisms like domain specific data scrubbing tools have to be employed to guarantee a sufficient data quality. In this context data auditing tools that are developed following our proposed domain driven approach can serve as a valuable part of a holistic data quality management system.

References

[1] David W. Aha. Generalizing from Case Studies: A Case Study. In *Proceedings of the Ninth International Conference on Machine Learning (MLC-92)*, 1992.

[2] Andreas Bauer and Holger Günzel, editors. *Data-Warehouse-Systeme*. dpunkt-Verlag, Heidelberg, 2001.

[3] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104. ACM Press, 2000.

[4] Carla E. Brodley and Mark A. Friedl. Identifying Mislabeled Training Data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[5] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, 1997.

[6] Tamraparni Dasu and Theodore Johnson. Hunting of the Snark - Finding Data Glitches using Data Mining Methods. In *Proc. of the 1999 Conf. on Information Quality, MIT*, 1999.

[7] Larry P. English. *Improving Data Warehouse and Business Information Quality*. Wiley & Sons, New York, 1999.

[8] Usama M. Fayyad. Data Mining and Knowledge Discovery: Making Sense Out of Data. *IEEE Expert*, 11(5):20–25, 1996.

[9] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. AJAX: An extensible data cleaning tool. *ACM SIGMOD Record*, 29(2):590, 2000.

[10] Udo Grimmer and Holger Hinrichs. A Methodological Approach To Data Quality Management Supported by Data Mining. In *Proceedings of the 6th International Conference on Information Quality (IQ 2001)*, pages 217–232, 2001.

[11] M. A. Hernandez and S. J. Stolfo. The Merge / Purge Problem for Large Databases. In *Proc. of the 1995 ACM SIGMOD Conf.*, 1995.

[12] H. Hinrichs. Datenqualitätsmanagement in Data Warehouse–Umgebungen. In A. Heuer, F. Leymann, and D. Priebe, editors, *Datenbanksysteme in Büro, Technik und Wissenschaft, 9. GI–Fachtagung BTW 2001, Oldenburg*, pages 187–206, Berlin, März 2001. Springer.

[13] H. Hinrichs and T. Wilkens. Metadata-Based Data Auditing. In N. F. F. Ebecken and C. A. Brebbia, editors, *Data Mining II (Proc. of the 2nd Intl. Conf. on Data Mining, Cambridge, United Kingdom)*, pages 141–150, Southampton, 2000. WIT Press.

[14] Jochen Hipp, Ulrich Güntzer, and Udo Grimmer. Data Quality Mining – Making a Virtue of Necessity. In *Proc. of the 6th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2001)*, pages 52–57, Santa Barbara, California, 2001.

[15] K.-T. Huang, Y. W. Lee, and R. Y. Wang. *Quality Information and Knowledge Management*. Prentice Hall, New Jersey, 1998.

[16] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and Quality in Data Warehouses. In *Proc. of the 10th Intl. Conf. CAiSE*98, Pisa, Italy*, Berlin, 1998. Springer.

[17] M.A. Jeusfeld, C. Quix, and M. Jarke. Design and Analysis of Quality Information for Data Warehouses. In *Proc. of the 17th International Conference on the Entity Relationship Approach (ER'98)*, Singapore, 1998.

[18] B.K. Kahn and D.M. Strong. Product and Service Performance Model for Information Quality: An Update. In *Proceedings of the 1998 Conference on Information Quality*, 1998.

[19] Jonathan I. Maletic and Andrian Marcus. Data cleansing: Beyond integrity analysis. In *Proceedings of The Conference on Information Quality (IQ2000)*, pages 200–209, MIT, Boston, October 2000.

[20] Meta Group. *Data Warehouse Scorecard*. Meta Group, 1999.

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

[22] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

[23] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10:334–350, 2001.

[24] C. Schaffer. A Conservation Law for Generalization Performance. In *Proceedings of the 11th International Conference on Machine Learning*, pages 259–265, Palo Alto, 1994. Morgan Kaufmann.

[25] Wang, R. A Product Perspective on Total Data Quality Management. *Communications of the ACM*, 41(2), 1998.