Cleaning Disguised Missing Data: A Heuristic Approach *

Ming Hua
School of Computing Science
Simon Fraser University, Canada
mhua@cs.sfu.ca

Jian Pei
School of Computing Science
Simon Fraser University, Canada
jpei@cs.sfu.ca

ABSTRACT

In some applications such as filling in a customer information form on the web, some missing values may not be explicitly represented as such, but instead appear as potentially valid data values. Such missing values are known as *disguised missing data*, which may impair the quality of data analysis severely, such as causing significant biases and misleading results in hypothesis tests, correlation analysis and regressions. The very limited previous studies on cleaning disguised missing data use outlier mining and distribution anomaly detection. They highly rely on domain background knowledge in specific applications and may not work well for the cases where the disguise values are inliers.

To tackle the problem of cleaning disguised missing data, in this paper, we first model the distribution of disguised missing data, and propose the embedded unbiased sample heuristic. Then, we develop an effective and efficient method to identify the frequently used disguise values which capture the major body of the disguised missing data. Our method does not require any domain background knowledge to find the suspicious disguise values. We report an empirical evaluation using real data sets, which shows that our method is effective – the frequently used disguise values found by our method match the values identified by the domain experts nicely. Our method is also efficient and scalable for processing large data sets.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms, Design, Experimentation

Keywords

Data Quality, Data Cleaning, Disguised Missing Data

*This research is supported in part by NSERC Discovery Grant 312194-05. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

KDD'07, August 12–15, 2007, San Jose, California, USA.

1. INTRODUCTION

Processing missing values is one of the most important tasks in data cleaning. Many methods have been developed to handle explicitly missing values or conduct analysis and data mining on noisy data sets with explicitly missing data.

Interestingly, in many applications, some missing values may not be explicitly represented as such, but instead appear as potentially valid data values. Such missing values are known as *disguised missing data* [5].

EXAMPLE 1 (DISGUISED MISSING VALUES). Consider the situation where a customer fills in an online application form of a frequent flyer program. Attribute `gender` has two choices: `male` or `female`. A system may set one of the two values, say `male` in this example, as the default value. Many customers may not want to disclose this information, or may not want to spend time to fill in the information. The consequence is that many missing values may disguise themselves as the default value, `male` in this case.

Using system default values is not the only cause translating to disguised missing data. As another example, the attribute `birth date` is often required in many customer account registration forms. However, many customers do not want to disclose their privacy. Popularly, one may choose January 1 (the first value in the pop-up lists of month and day, respectively) in order to pass. Here, January 1 is a disguise for the missing data. ■

Disguised missing data exist in real applications. For example, we will analyze two real data sets in Section 5: the Pima Indians Diabetes database containing records about Pima Indian females who are at least 21 years old and tested either positive or negative for diabetes, and the Adverse Event Reporting System data set from the U.S. Food and Drug Administration in the first quarter of 2004. Disguised missing data are detected in both data sets. Interestingly, many previous machine learning studies (e.g., [2, 6]) use the Pima Indians Diabetes database but presume that the data set has no missing data.

Disguised missing data may impair the quality of data analysis severely. Significant biases may be caused by the disguised missing data. As illustrated in [5], due to the disguised missing data, some simple statistics such as standard deviation may shift to some anomalous values. Moreover, hypothesis tests, correlation analysis and regressions using disguised missing data may give misleading results.

Disguised missing data pose a much more serious challenge for data cleaning than explicitly missing values. For explicitly missing values, the exact missing entries are known

and thus strategies can be developed to avoid using those entries in data analysis. For disguised missing data, however, we may not even know the exact missing entries. In the situations illustrated in Example 1, the resulting data set may contain the male customers who did provide the information, and some customers born on January 1. How to distinguish those disguised missing values and those real values is far from trivial.

Although there are extensive studies on data analysis with missing values, to the best of our knowledge, the problem of cleaning disguised missing data has not been investigated thoroughly. In this paper, we tackle the problem and make the following contributions. First, we analyze the distribution of disguised missing values and identify the important and interesting *embedded unbiased sample* (EUS) heuristic that often holds for disguised missing values: the projected database of a disguise value often contains a large unbiased sample of the whole data set. Based on this property, we propose a general framework to identify suspicious frequently used disguise values. Second, mining frequently used disguise values from large data sets is computationally challenging. We devise efficient and scalable heuristic algorithms. Last, we test our approach using both real data sets and synthetic data sets. The experimental results show that our method is effective—the frequently used disguise values found by our method match the values identified by the domain experts nicely. Our method is also efficient and scalable for processing large data sets.

The rest of the paper is organized as follows. In Section 2, we describe disguised missing data formally and review related work. In Section 3, we present the embedded unbiased sample property and propose a framework of finding suspicious frequently used disguise values. We develop an efficient data mining approach in Section 4. Experimental results are reported in Section 5. The paper is concluded in Section 6.

2. DISGUISED MISSING DATA

For a tuple t in a table T, the value of t on attribute A is denoted by $t.A$, which is also called an *entry*. In data collection, for an entry $t.A$, three situations may arise.

Case 1: The user provides a value to the entry that, to the best of the user's knowledge, reflects the fact in the real world and should be captured. Due to observation errors, it is possible that a user provides an incorrect value to the entry. The point we want to make in this case is that *the entry value is not missing in its nature.*

Case 2: The user does not provide a value. In other words, $t.A$ is *explicitly missing*, denoted by $t.A = \otimes$, where $\otimes$ is a meta symbol not in the domain of any attribute.

Case 3: The user does not intend to provide a value that reflects the fact in the real world. However, due to some data collection mistakes, such as the cases illustrated in Example 1, a value in the domain of A is recorded in the table. In other words, a *disguised missing value* happens. Although the entry value is missing in its nature, the table records a "fake" value, which is called a disguise value.

Formally, let T be the *truth table* and $\widetilde{T}$ be the *recorded table*. Here, the truth table contains the data that should be recorded. There exists a one-to-one mapping between the tuples in T and $\widetilde{T}$. That is, for any tuple $t \in T$, there is a corresponding tuple $\widetilde{t} \in \widetilde{T}$, and vice versa.

For any entry $t.A$, if $t.A \neq \otimes$, then $\widetilde{t}.A = t.A$. That is, if an entry in the truth table is not missing, then it is collected correctly in the recorded table. However, if $t.A = \otimes$, then $\widetilde{t}.A$ can be either $\otimes$ or a legal value in the domain of A. Particularly, an entry $\widetilde{t}.A$ is called *disguised missing* if $t.A = \otimes$ but $\widetilde{t}.A \neq \otimes$. The value $\widetilde{t}.A$ is the *disguise* of the disguised missing entry, or called a *disguise value*. Figure 1 summarizes some frequently used notations in the paper.

Symbol	Explanation
T	The truth table
$\widetilde{T}$	The recorded table
$\widetilde{T'}$	A subset of $\widetilde{T}$
t, $\widetilde{t}$	A tuple in the fact/recorded table
$t.A$, $\widetilde{t}.A$	An entry in the fact/recorded table
$\widetilde{T}_v$	The projected database of value v
S_v	The disguised missing set of v
M_v	The maximal embedded unbiased sample of v
$\phi(\widetilde{T}, \widetilde{T'})$	The correlation-based sample quality score

Figure 1: Some frequently used notations.

In data cleaning, we are given the recorded table $\widetilde{T}$, while the truth table T is typically unavailable. Ideally, the problem of *cleaning disguised missing data* is to find the values that are frequently used as disguise values, and the set of disguised missing entries.

Cleaning disguised missing data in general is very difficult. As an extreme example, if the missing values in the truth table are disguised by independent and random values in the domain of the attribute, it is very hard to unmask them without any hints.

Several heuristic approaches have been considered for cleaning disguised missing data. For example, with background knowledge, a domain expert can screen entries with suspicious values, such as blood pressure of 0. More generally, outliers can be found and considered as potential disguised missing values [4]. However, if a data set has many disguised missing entries, such as those disguised `male` entries in Example 1, they may not appear as outliers. One heuristic way to detect the existence of such disguised missing data is to detect distribution anomalies [4]. For example, if we observe that the number of tuples having value `male` is much larger than that of tuples having value `female`, and we know that the populations of males and females in the data set should be balanced, then we can conclude that some `male` entries in the data set may be disguised missing.

Although the problem of cleaning disguised missing data has been tackled from some angles, the existing approaches often rely on domain knowledge heavily, and are developed for specific applications. However, domain knowledge is often incomplete or even unavailable for many data analysis tasks. For example, the correct distribution of an attribute may not be known. Then, analyzing suspicious values, outliers and distribution may become difficult and unreliable. As observed in [1], a missing value may disguise itself as an *inlier*, a data value that lies in the interior of the statistical distribution of normal data values. Those inlier disguised missing values cannot be detected effectively using the existing methods. In addition, due to their heavy reliance on domain knowledge, most of the existing methods are not general and capable for generic applications.

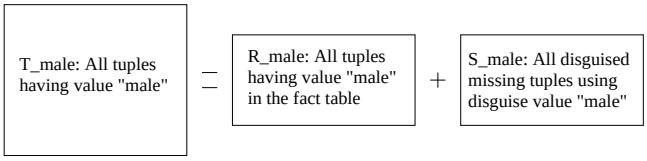

Figure 2: The EUS heuristic.

3. GENERAL FRAMEWORK

In this section, we first observe the embedded unbiased sample heuristic of frequent disguise values. Then, we propose a framework for cleaning disguised missing data, and analyze the computational challenges.

3.1 Embedded Unbiased Sample Heuristic

As analyzed before, if missing values disguise themselves randomly, it is very difficult to identify the disguised missing data. Fortunately, such random disguising often does not happen extensively in practice. Instead, as illustrated in Example 1, a small number of values (typically one or two in an attribute) are frequently used as the disguises. It is practical to make the following assumption.

ASSUMPTION 1 (FREQUENTLY USED DISGUISES). *On an attribute, there often exist only a small number of disguises that are frequently used by the disguised missing data.* ■

Under the missing completely at random (MCAR) and missing at random (MAR) models [3], missing data are often distributed randomly in real data sets. Consequently, disguised missing entries are often distributed randomly, too, as verified by our experimental results using real data sets.

EXAMPLE 2 (EUS HEURISTIC). Consider the situation described in Example 1. Let $\widetilde{T}_{\texttt{male}}$ be the set of tuples carrying value `male` on attribute `gender`. Conceptually, $\widetilde{T}_{\texttt{male}}$ can be divided into two exclusive subsets as shown in Figure 2.

The first subset, $R_{\texttt{male}}$, contains those tuples whose values on attribute `gender` are not missing in the truth table. The second subset, $S_{\texttt{male}}$, contains those tuples whose values on attribute `gender` are disguised missing and the value `male` is used by them as the disguise value.

Heuristically, if the disguised entries are randomly distributed and value `male` is frequently used as a disguise, the set $S_{\texttt{male}}$ is an unbiased sample of the truth table except for attribute `gender` itself (all tuples in $S_{\texttt{male}}$ take value `male` on `gender`). Similarly, we can also divide $\widetilde{T}_{\texttt{female}}$, the set of tuples having value `female` on attribute `gender`, into two subsets $R_{\texttt{female}}$ and $S_{\texttt{female}}$. If value `male` is used more frequently as the disguise value on attribute `gender`, then $S_{\texttt{male}}$ from $\widetilde{T}_{\texttt{male}}$ is larger than $S_{\texttt{female}}$ from $\widetilde{T}_{\texttt{female}}$.

According to Assumption 1, on each attribute, there are only a very small number of values that are used as disguises. In other words, it is likely those disguise values contain subsets of tuples that are unbiased samples of the whole data set. As a heuristic, if a value contains a large subset of tuples that is an unbiased sample of the whole data set, this value is suspicious of a disguise value. ■

For a value v on attribute A, the set of tuples in $\widetilde{T}$ carrying the value on the attribute is called the *projected database* of v, denoted by $\widetilde{T}_{A=v} = \{\widetilde{t} \in \widetilde{T} | \widetilde{t}.A = v\}$. Hereafter, for the sake of brevity, we assume that the domains of attributes are exclusive, and thus a value belongs to the domain of at most one attribute. Then, we can write $\widetilde{T}_{A=v}$ as $\widetilde{T}_v$.

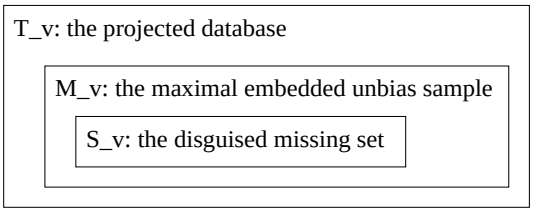

Figure 3: The relationship among several concepts.

We observe the following *embedded unbiased sample heuristic* (*EUS heuristic* for short) of disguised missing data.

The Embedded Unbiased Sample Heuristic *If v is a frequently used disguise value on attribute A, then $\widetilde{T}_{A=v}$ contains a large subset $S_v \subseteq \widetilde{T}_{A=v}$ such that S_v is an unbiased sample of $\widetilde{T}$ except for attribute A.* ■

Interestingly, the heuristic is also applicable to continuous attributes and does not need discretization, since as the frequently used disguise values are fixed, their projected databases may often be much larger than the projected databases of other values. It is insensitive to the variances of attribute values due to the same reason.

3.2 A General Framework

Since a small number of values may be used frequently as disguises, a critical step in cleaning disguised missing data is to *find the disguise values used frequently in attributes*.

For each value v on attribute A, let T_v be the set of tuples carrying value v in the truth table. Clearly, $T_v \subseteq \widetilde{T}_v$. Then, $S_v = (\widetilde{T}_v - T_v)$ is the set of tuples using v as the disguise on attribute A. We call S_v the *disguised missing set* of v.

According to the EUS heuristic, S_v is an unbiased sample of $\widetilde{T}$. The larger the size of S_v, the more frequently v is used as the disguise value. A value v is called a *frequent disguise value* if it is frequently used as disguises.

Unfortunately, S_v is unknown and cannot be computed accurately from $\widetilde{T}$ in general. The EUS heuristic suggests a heuristic way to find those frequent disguise values. Essentially, on each attribute, we can find a small number of attribute values whose projected databases contain a large subset as an unbiased sample of the whole table. Such attribute values are suspects of frequently used disguise values. The larger the unbiased sample subset, the more likely the value is a disguise value.

Technically, for value v on attribute A, let $M_v \subseteq \widetilde{T}_v$ be the maximal subset of $\widetilde{T}_v$ that is an unbiased sample of D. M_v is called the *maximal embedded unbiased sample*, or *MEUS* for short. We can use the size and the quality (i.e., how well it resembles the distribution of the whole data set) of M_v as the indicators of how likely that v is a disguise value. Those values with a large MEUS should be reported as the suspects of frequent disguise values. Figure 3 illustrates the relationship among the three sets $\widetilde{T}_v$, M_v, and S_v.

On the other hand, the above heuristic may not hold all the time. For example, many people may submit their tax returns on the deadline day. Thus, on attribute `submission-date`, the projected databases of the dates on or right before

Phase 1: Mining candidates of frequent disguise values

Input: A table T and a threshold on the number of candidates of frequent disguise values k;

Output: for each attribute, k candidates of frequent disguise values;

Method:

1: `FOR` each attribute A `DO`
2: // applicability test
check whether the projected databases of most (frequent) values on A are unbiased samples of T, if so, `break`;
3: `FOR` each value v on A `DO` derive M_v;
4: find the value(s) with the best and largest M_v's;
`END FOR`

Phase 2: postprocessing: verify the candidates of frequent disguise values;

Figure 4: The framework.

the deadline may likely contain large unbiased samples of all tax returns, while the dates on or right before the deadline are often not frequently used disguise values.

To ensure that the EUS heuristic fits a data set, before we apply the heuristic approach to look for suspicious disguise values in an attribute, we should first test whether most of the projected databases of the frequent values in the attribute are unbiased samples of the whole database. The heuristic should be applied only if most of the projected databases are not unbiased samples.

Based on the above discussion, a general framework of cleaning disguised missing data is shown in Figure 4. The framework is in two phases. In the mining phase, we analyze each attribute to check whether our heuristic approach is applicable. We find the candidates of frequent disguise values on the applicable attributes. In the postprocessing phase, those candidates can be verified by domain experts or other data cleaning methods.

In the rest of this paper, we focus on the first phase of the framework, which reduces the number of candidates substantially and thus makes the domain experts' analysis effective. Please note that in our approach, the first phase does not require any domain knowledge and can be conducted automatically.

4. A DATA MINING APPROACH

There are some important technical challenges.

First, *how can we measure whether a set of tuples is an unbiased sample of a table?* Typically, the table in question is of multiple attributes. A sample with a non-trivial sample rate (i.e., not close to 100%) may lose some information about the rare combinations of attribute values. Statistically, measuring whether two multidimensional data sets have a similar distribution is far from trivial.

Second, *how can we compute a maximal embedded unbiased sample M_v from the projected database $\widetilde{T}_v$?* One serious difficulty is that the subsets are not monotonic in terms of their similarity to the whole data set in distribution. For $U \subseteq \widetilde{T}_v$, U may contain a subset $V \subset U$ such that V resembles the distribution of $\widetilde{T}$ better than U. On the other hand, another subset $W \subset U$ may be worse than U in terms of being an unbiased sample. In general, finding M_v is computationally costly.

Last, *Can we avoid computing the MEUS for every attribute value v?* As computing the MEUS for a value is already costly, computing all MEUS's for all values on all attributes can be too expensive in large databases where there are tens of attributes and each attribute has tens or hundreds of possible values on average.

4.1 Correlation-Based Sample Quality Score

Given table $\widetilde{T}$ on attributes $A_1, \ldots, A_n$ and a subset $\widetilde{T'} \subset \widetilde{T}$, we want to measure whether $\widetilde{T'}$ is a good sample of $\widetilde{T}$. Intuitively, correlations can capture the distribution of a data set nicely. If values correlated in $\widetilde{T}$ are also correlated in $\widetilde{T'}$ and vice versa, then likely $\widetilde{T}$ and $\widetilde{T'}$ are of similar distribution. Based on this intuition, we can use correlations of pairs of values to measure how good a sample $\widetilde{T'}$ is with respect to $\widetilde{T}$.

Technically, let $v_{i_1}, \ldots, v_{i_l}$ be values on attributes $A_{i_1}, \ldots, A_{i_l}$, respectively, where $A_{i_j} \neq A_{i_k}$ for $j \neq k$. The probability of a tuple in $\widetilde{T}$ having $v_{i_1}, \ldots, v_{i_l}$ is given by

$$P_{\widetilde{T}}(v_{i_1}, \ldots, v_{i_l}) = \frac{|\{\widetilde{t} \in \widetilde{T}| \wedge_{j=1}^{l} (\widetilde{t}.A_{i_j} = v_{i_j})\}|}{|\widetilde{T}|}.$$

The *correlation* among the occurrences of $v_{i_1}, \ldots, v_{i_l}$ is measured by

$$Corr_{v_{i_1}, \ldots, v_{i_l}} = \frac{P_{\widetilde{T}}(v_{i_1}, \ldots, v_{i_l})}{\Pi_{j=1}^{l} P_{\widetilde{T}}(v_{i_j})}.$$

Similarly, we can obtain the probability and the correlation on subset $\widetilde{T'}$.

Computing correlations on multiple variables can be costly on large data sets. Practically, we can consider only correlations of variable pairs. Let v_i and v_j be two values on attributes A_i and A_j, respectively. The *correlation* between v_i and v_j (also called the *lift* sometimes) is given by

$$Corr(v_i, v_j) = \frac{P(v_i, v_j)}{P(v_i)P(v_j)} = \frac{P(v_j|v_i)}{P(v_j)}.$$

To measure how good a sample the subset $\widetilde{T'}$ is with respect to $\widetilde{T}$, we compare the correlations in $\widetilde{T}$ and $\widetilde{T'}$ and calculate the *correlation-based sample quality score*, denoted by $\phi(\widetilde{T}, \widetilde{T'})$, as

$$\sum_{P_{\widetilde{T'}}(v_i, v_j) > 0} \frac{P_{\widetilde{T}}(v_i, v_j)}{1 + |Corr_{\widetilde{T}}(v_i, v_j) - Corr_{\widetilde{T'}}(v_i, v_j)|^q}, \quad (1)$$

where $Corr_{\widetilde{T}}$ and $Corr_{\widetilde{T'}}$ are the correlations on $\widetilde{T}$ and $\widetilde{T'}$, respectively, and q is the order imitating the order of Minkowski distances.

The correlation-based sample quality score uses the value pairs in the subset $\widetilde{T'}$ as features to measure the sample quality. A sample quality score is a non-negative number. The larger the score, the better $\widetilde{T'}$ an unbiased sample of $\widetilde{T}$. In the score, we accommodate the following major factors.

First, the score opts for subsets $\widetilde{T'}$ where the correlations in $\widetilde{T'}$ are similar to those in $\widetilde{T}$. However, since $\widetilde{T'}$ is a sample of $\widetilde{T}$ and is smaller in size, there can be many pairs of attribute values appearing in $\widetilde{T}$ but not in $\widetilde{T'}$. Thus, the score

does not check pairs correlated in $\widetilde{T}$ but not appearing in $\widetilde{T'}$. Instead, to measure whether those popular correlations are captured, the probability of an attribute value pair (i.e., $P(v_i, v_j)$ as the numerator) is used as the weight. A sample capturing more correlations and more popular correlations has a higher score.

Second, in the score, we use correlations of attribute value pairs. Generally, we can use correlations of sets of attribute values of arbitrary size. In the worst case, two data sets may have very similar correlations on every pair of attribute values, but may still be quite different in the full space distribution. However, computing correlations among multiple attribute values can be expensive on large data sets. On the other hand, in practice, correlations of attribute value pairs are often good enough to serve the purpose of measuring the similarity of distributions, as shown by our experimental results on real data sets.

Last, we normalize the difference between the correlations of an attribute value pair in $\widetilde{T}$ and $\widetilde{T'}$ to range $[0, 1]$ by using the difference of correlations in the denominator. This technical treatment avoids biasing value pairs of very similar correlations in $\widetilde{T}$ and $\widetilde{T'}$.

For each value v on attribute A, we compute M_v, the MEUS of v. To measure the potential that a value v is a frequent disguise value, we consider two aspects: the quality of the MEUS measured by $\phi(\widetilde{T}_v, M_v)$ and the relative size of M_v with respect to $\widetilde{T}_v$. The latter aspect reflects how well the projected database fits the EUS heuristic. Thus, we define the *disguise value score* (DV-score for short) of a subset $U \subseteq \widetilde{T}_v$ as $dv(v, U) = \frac{|U|}{|\widetilde{T}_v|}\phi(\widetilde{T}_v, U)$. Moreover, the *(frequent) disguise value score*(*DV-score* for short) of v is defined as

$$dv(v) = \max_{U \subseteq \widetilde{T}_v}\{dv(v, U)\} = \max_{U \subseteq \widetilde{T}_v}\{\frac{|U|}{|\widetilde{T}_v|}\phi(\widetilde{T}_v, U)\}.$$

M_v, the maximal embedded unbiased sample, is a subset maximizing the DV-score. That is,

$$M_v = \arg\max_{U \subseteq \widetilde{T}_v}\{\frac{|U|}{|\widetilde{T}_v|}\phi(\widetilde{T}_v, U)\}.$$

4.2 Finding Approximate MEUS's

Generally, the DV-score is not monotonic with respect to the set containment relation. That is, for a subset $U \subseteq \widetilde{T}_v$ and $W \subset U$, $dv(v, W)$ may be larger or smaller than $dv(v, U)$. The non-monotonic nature of the DV-score indicates that computing the maximal embedded unbiased samples is computationally challenging. In the worst case, one may have to consider an exponential number of subsets in order to find M_v. It is often too costly for popular values (i.e., $|\widetilde{T}_v|$ is large) in large databases.

To tackle the problem practically, we adopt a greedy approach as shown in Figure 5. We start with the projected database $\widetilde{T}_v$ as the initial sample. In each interaction, for a tuple $\widetilde{t}$ in the current sample, we calculate the DV-score gain if $\widetilde{t}$ is removed from the current sample set. A tuple with the largest positive DV-score gain is removed as the result of the current iteration. The iteration continues until the DV-score cannot be improved further by removing one tuple from the current sample. The resulting sample is output as the approximation of M_v.

Input: a table T and a value v on attribute A;
Output: approximate M_v;
Method:
1: $U \leftarrow \widetilde{T}_v$;
2: `REPEAT`
3: `FOR EACH` tuple $\widetilde{t} \in U$
compute the DV-score gain of $(U - \{\widetilde{t}\})$ over U;
4: remove a tuple $\widetilde{t}_0$ with the largest DV-score gain if the gain is positive;
5: `UNTIL` no tuple can be removed;
6: `RETURN` U;

Figure 5: A greedy method to compute approximate MEUS.

4.3 Efficient Implementation

A straightforward implementation of the greedy algorithm may still be costly on large databases. Once a tuple is removed, the total number of tuples in the current sample is reduced, and thus the correlation between every value pair changes. Therefore, for each round in the iteration, we may have to update the correlation of each value pair, and also recompute the DV-score gain for every surviving tuple in the current sample. The complexity of the algorithm is $O(|\widetilde{T}_v|^2)$. Clearly, when the projected database $\widetilde{T}_v$ is large, the overhead can be substantial. Here, we present some techniques to improve the efficiency of the greedy search.

4.3.1 Searching Using Contribution Scores

Since the DV-score of a tuple changes whenever the sample changes, and computing the DV-score gain can be costly, we would like to find some heuristics that help us to, without re-computing the DV-scores for all tuples, identify the tuples that may lead to good DV-score gains.

Let *support* $sup(v_i)$ be the number of tuples in the current sample that contain value v_i. We denote the number of tuples in the current sample by n. Then, the correlation of v_i and v_j in the current sample can be rewritten as $Corr(v_i, v_j) = \frac{sup(v_i, v_j)}{sup(v_i)sup(v_j)}n$.

Once a tuple is removed, one of the following three cases arises.

Case 1: If the tuple does not contain v_i or v_j, then $sup(v_i, v_j)$, $sup(v_i)$ and $sup(v_j)$ remain, but n decreases. The correlation decreases, too. The change is $\frac{sup(v_i, v_j)}{sup(v_i)sup(v_j)}$.

Case 2: If the tuple contains both v_i and v_j, removing the tuple leads to a change of the correlation between v_i and v_j. However, how the correlation changes depends on the values of the supports of v_i, v_j and their combination.

Case 3: If the tuple contains only v_i or v_j but not both, then removing the tuple boosts the correlation between v_i and v_j. The change is $\frac{n - sup(v_i)}{sup(v_i) - 1} \cdot \frac{sup(v_i, v_j)}{sup(v_i)sup(v_j)}$ if the tuple contains v_i but not v_j; and is $\frac{n - sup(v_j)}{sup(v_j) - 1} \cdot \frac{sup(v_i, v_j)}{sup(v_i)sup(v_j)}$ if the tuple contains v_j but not v_i.

Based on the above observations, we can maintain the correlation information as follows. For each value pair (v_i, v_j), we keep its correlation and its probability in the whole data set, i.e., $Corr_{\widetilde{T}}(v_i, v_j)$ and $P_{\widetilde{T}}(v_i, v_j)$. Those values do not change during the computation and thus should be computed only once.

We also maintain the correlation between v_i and v_j in the current sample, i.e., $Corr_U(v_i, v_j)$, and whether it is greater than or smaller than $Corr_{\widetilde{T}}(v_i, v_j)$. Instead of computing the DV-score gain for each tuple at each iteration, which is very costly, we assign each tuple $\widetilde{t}$ a *contribution score* as $c(\widetilde{t}) = \sum_{v_i, v_j \in \widetilde{t}, P_U(v_i, v_j) > 0} P_{\widetilde{T}}(v_i, v_j) f(v_i, v_j, \widetilde{t})$, where $f(v_i, v_j, \widetilde{t}) = 1$ if removing $\widetilde{t}$ reduces the difference between $Corr_U(v_i, v_j)$ and $Corr_{\widetilde{T}}(v_i, v_j)$; and, otherwise, -1.

The value of f can be computed quickly using the three cases of how $Corr(v_i, v_j)$ changes as analyzed before. Once a tuple is removed, we update the correlations affected. Then, a surviving tuple in the current sample changes its contribution score only if it contains some value pairs whose correlations are affected. We do not need to recompute the contribution scores of other tuples. Comparing to computing DV-score gains, contribution scores are often much easier to maintain.

Following the spirit of the greedy search, we pick the tuple with the largest positive contribution score, compute its DV-score gain, and remove it from the current sample if the gain is positive. The algorithm terminates if we cannot find any tuple with a positive contribution score, or the DV-score gain is not positive. As shown in our experiments using real data sets, the heuristic method improves the search efficiency substantially, and still gets results of good quality.

4.3.2 Pruning Other Values in the Same Attribute

Computing the approximate MEUS for a value v may not be cheap. If we have to compute the approximate MEUS for each value on an attribute, it can be expensive. Once we compute the MEUS for one value or some MEUS's for several values on one attribute, can we use the information to prune the computation of the MEUS's of other values on the same attribute? There are multiple values on an attribute. In which order should we process those values and derive their DV-scores so that more pruning may happen?

If we can get large DV-scores from the first several values that we process, then we may have a better opportunity to prune the remaining values. Based on this observation, we should start with the most frequent value on an attribute. Heuristically, a large projected database may have a better chance to have a subset as an unbiased sample, and thus achieve a better DV-score. Therefore, *on each attribute, we should process the values in their support descending order to compute their DV-scores.*

Suppose u is the value on attribute A that has the largest DV-score so far. When we consider value w on the same attribute, if we can determine that $dv(w)$ must be smaller than $dv(u)$, then we do not need to compute the exact value of $dv(w)$, since u is a better suspect of frequent disguise value on this attribute. *How can we determine early that $dv(w)$ is smaller than $dv(u)$?*

If U is the current sample of w in the greedy search (Figure 5), then, since in the definition of the correlation-based sample quality score (Equation 1)

$$\frac{1}{1 + |Corr_{\widetilde{T}}(v_i, v_j) - Corr_{\widetilde{T}'}(v_i, v_j)|^q} \leq 1,$$

an upper bound of $d(w, U)$ is given by

$$\begin{aligned} d(w, U) &\leq \frac{|U|}{|\widetilde{T}_v|} \sum_{P_U(v_i, v_j) > 0} P_T(v_i, v_j) \\ &\leq \sum_{P_U(v_i, v_j) > 0} P_T(v_i, v_j). \end{aligned} \quad (2)$$

Therefore, during the greedy search, we check after each iteration $\frac{|U|}{|\widetilde{T}_v|} \sum_{P_U(v_i, v_j) > 0} P_T(v_i, v_j)$ against $dv(u)$. Once $\frac{|U|}{|\widetilde{T}_v|} \sum_{P_U(v_i, v_j) > 0} P_T(v_i, v_j)$ becomes smaller than $dv(u)$, the search should stop since v cannot have a better DV-score than u due to the first inequality of Equation 2.

Furthermore, for a value v, if $\sum_{P_U(v_i, v_j) > 0} P_T(v_i, v_j)$ is smaller than $dv(u)$, then we even do not need to search for v according to the second inequality of Equation 2. For many infrequent values on the attribute, this case may happen and thus those infrequent values can be pruned directly.

5. EXPERIMENTAL RESULTS

In this section, we report a systematic empirical study using real data sets and synthetic data sets. All the experiments were conducted on a PC computer running the Microsoft Windows XP Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk. Our algorithms were implemented in Microsoft Visual C++ V6.0. By default, our method was implemented as described in Section 4. We used the greedy search method with the help of contribution score, and set the order $q = 1$ in the correlation-based sample quality score (Equation 1).

5.1 Finding Suspicious Frequent Disguise Values on Real Data Sets

We tested our method on two real data sets, the Pima Indians Diabetes Database from the UCI Machine Learning Database Repository[1] and the AERS (Adverse Event Reporting System) data set from the U.S. Food and Drug Administration (FDA)[2]. In both data sets, all attributes pass the applicability test of our approach. That is, for each attribute there exists at least one relatively frequent value whose projected database is not an unbiased sample of the whole data set. We used some thresholds to conduct the test. Limited by space, we omit the details here.

In the rest of this section, our goal is to find the most likely frequent disguise value for each attribute.

There are 768 records in the Pima Indians Diabetes data set. All the records are on eight attributes of Pima Indian females who are at least 21 years old and tested either positive or negative for diabetes. The domain of each attribute is numeric, and no explicitly missing value is reported. However, some disguise missing data are detected by examining the suspicious values found in our experiments as shown in Table 1.

On the attributes *diastolic blood pressure*, *triceps skin fold thickness*, *2-hour serum insulin*, and *body mass index*, our method detects 0 as the most frequent disguise value. The results match the domain knowledge. In those attributes, 0 is a suspicious value since unlikely those attributes take values 0 for a person of reasonable condition. There are 35, 227, 374 and 11 tuples having value 0 in those four attributes, respectively. Each projected database of those values forms the maximum embedded unbiased sample of the whole database. In other words, those values are quite evenly distributed in the data set. This observation strongly supports the embedded unbiased sample property of the frequent disguise values.

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html.
[2] http://www.fda.gov/cder/aers/aers-prev-data.htm.

Attribute	Most frequent disguise value	Number of Occurrences	Number of tuples in the approximate MEUS
Number of times pregnant	0	111	110
Plasma glucose concentration at 2 hours	91	9	9
Diastolic blood pressure (mm Hg)	0	35	35
Triceps skin fold thickness (mm)	0	227	227
2-Hour serum insulin (mu U/ml)	0	374	374
Body mass index (weight in kg/(height in m)^2)	0	11	11
Diabetes pedigree function	no	no	no
Age (years)	21	63	57

Table 1: Suspicious Disguised Missing Values in the Pima Indians Diabetes data Set.

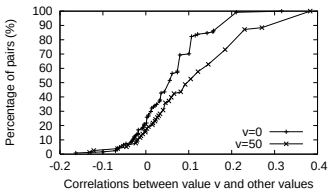

Figure 6: Distribution of covariance correlation between the frequent disguise value and values in other attributes.

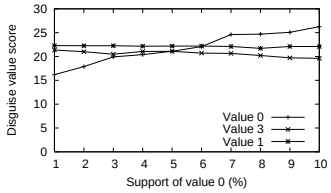

Figure 7: DV-score with respect to Support of outlier disguise value.

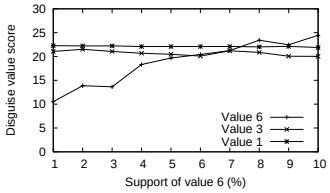

Figure 8: DV-score with respect to Support of outlier disguise value.

Let us further analyze the suspicious disguise value diastolic blood pressure 0 in detail. Figure 6 plots the covariance correlations between value 0 and values on other attributes. To make the figure easy to read, we plot the cumulative frequencies of the correlations. That is, we sort all values on other attributes according to their covariance correlations with respect to 0 on attribute *diastolic blood pressure*, in value ascending order. The figure plots the percentage of values whose covariance correlation with 0 are up to e, while e is the variable shown in the horizontal axis. More than 80% of the values have a covariance correlation in range $[-0.1, 0.1]$. This highly indicates that value 0 on this attribute is distributed evenly. For comparison, we also plot the distribution of normal value 50 as an example. As can be seen, value 50 has stronger correlations with values other than 0.

Our method is different from simply picking the most frequent value on an attribute as the suspect. For example, on attribute *diastolic blood pressure*, our method picks value 0, which turns out is a suspicious value. In the same attribute, value 70 is the most frequent one, appearing 57 times. In fact, value 70 corresponds to the normal blood pressure of a human being in good health. The reason that our method can pick 0 as a suspicious value instead of 70 is that the normal blood pressure may be correlated with some other attribute values for a person in good health, which makes the value 70 not evenly distributed in the database.

Our method can find not only disguise values that are outliers or suspects in the domains of the attributes, but also those inliers which cannot be found by previous methods. For example, on the attribute *age*, value 21 is picked by our method as a suspicious frequent disguise value. The domain of the attribute is from 21 to 81. However, value 21 appears in 63 tuples, nearly 10% of the data set. Our greedy method finds a sample of 57 tuples that resembles the distribution of the whole data set well. A conjecture here is that value 21 may be used as the default value when the patients' age information is collected.

On some attributes where the domain is highly diverse and the frequency of every value is very low, our method cannot find any suspicious frequent disguise values. For example, in the attribute *Diabetes pedigree function*, the frequencies of all values are below 1%. Therefore, no suspicious disguise value is reported. In our implementation, we used a support threshold to prune.

The Pima Indians Diabetes Data set is widely used in previous machine learning studies (e.g., [2, 6]). Most of the previous studies using this data set presume that the data set has no missing data. However, the above analysis strongly indicates that a substantial part of the tuples may have disguised missing data in the five attributes, namely *diastolic blood pressure*, *triceps skin fold thickness*, *2-hour serum insulin*, *body mass index*, and *age*. With the disguised missing data, the results of those analysis may not be accurate.

We also tested our method on the AERS data set, which contains adverse events reported to FDA from January 1, 2004 to March 31, 2004. Since some attributes contain a large portion of explicitly missing values, we pick the 7 attributes with the fewest missing values, namely *ISR*, *CASE*, *Event_DT*, *FDA_DT*, *AGE*, *WT*, and *Rept_DT*, and remove the tuples with missing entries. There are totally $18,174$ records in the resulting data set.

Our method detects some inlier suspicious frequent disguise values, such as value *20030101* (i.e., Jan 1, 2003) on attribute *Event_DT*, value *20040319* on attribute *FDA_DT*, and value *20040311* on attribute *Rept_DT*. Our results match the analysis reported in [5], which identifies the suspicious disguise values using domain knowledge to detect abnormal statistic distributions. Different from [5], our method detects those disguise values without any domain knowledge.

The results on the two real data sets highly suggest that our method is effective and accurate in finding suspicious frequent disguise values. Moreover, our method can be applied to a wide range of applications, and does not need to use any domain knowledge to identify suspects. It can find frequent disguise values as outliers or inliers.

5.2 Detecting Disguised Missing Entries

We also tested whether our method can be used to detect not only the frequent disguise values, but also the disguised missing entries, that is, the exact locations (the tuples and the attributes) where the entries are disguised missing.

To the best of our knowledge, there are not real data sets in which the disguised missing entries are annotated. Thus, we modified a real data set in our test.

We used the Breast Cancer Wisconsin data set from the UCI Machine Learning Database Repository[3]. Totally there are 699 records in the data set. Each record represents one breast cancer case. There are only 16 missing values reported in the data set.

We randomly injected the disguised missing data into the attribute *Uniformity of Cell Size* whose domain is integers between 1 and 10 (inclusive). We chose this attribute because there is one very frequent value in this attribute, value 1 occurring in 384 tuples (i.e., 54.94% of the tuples in the data set). We want to test how sensitive our method is with respect to the population of disguised missing data.

We tested the capability of our method in detecting disguised missing values from two aspects.

First, we tested how well our method can identify the disguised missing values which use an outlier as the disguise. We randomly chose s% of the tuples in the data set and replaced their values on the *Uniformity of Cell Size* attribute by 0, where s is a parameter varying from 1% to 10% in our experiments. Figure 7 shows the DV-score of the outlier disguise value 0 and those of the top-2 legal values in the domain with the highest DV-scores.

When the support of 0 is greater than 6%, the DV-score of 0 is the highest on the attribute and thus 0 is captured by our method as the frequent disguise value. Although here the support of 1 is much higher than the support of 0 in such a case, our method can detect the disguise value accurately based on the distribution, and is robust to the large difference in frequency.

When the support of 0 is smaller than 6%, the DV-score of 0 cannot exceed those of 1 and 3, the two legal values with the highest DV-scores. In such a situation, the projected database of 0 is too small to resemble the distribution of the whole data set, and thus 0 cannot be detected by our method as a frequent disguise value.

This experiment clearly shows that our method can detect outlier disguise values in very low support, and is not sensitive to popular but not disguise values.

In the above case, the projected database of 0 is the MEUS returned by our method. Thus, once 0 is detected as the disguise value, all disguised missing entries can be identified correctly.

Second, we tested the effectiveness of our method on disguised missing entry detection for disguise values as inliers. We chose value 6 as disguise since it has the lowest DV-score in the attribute, which means that its projected database is the most biased. We randomly chose s% of tuples in the data set and replaced their values on the *Uniformity of Cell Size* attribute by 6, where s is a parameter varying from 1% to 10% in our experiments. The results are shown in Figure 8. When the support of 6 is 8% or over, value 6 has the largest DV-score and thus is identified by our method as the top suspicious frequent disguise value. When the support of 6 is 8%, there are 82 tuples in the projected database of 6, 56 of them are disguised missing values injected by us. Our method reports a MEUS of 6 having 75 tuples. Among those tuples, 55 are those injected by us. In other words, the MEUS returned by our method captures most of the disguised missing data. This also verifies the effectiveness of our greedy method in searching MEUS's.

From this set of experiments, we can clearly see that our method is capable of finding MEUS's for disguise values as either outliers or inliers. The MEUS's cover most of the disguised missing data. This property is important and very useful for cleaning disguised missing data.

5.3 Effectiveness of the Correlation-Based Sample Quality Score

The correlation-based sample quality score (Equation 1) takes a parameter, order q, to imitate the order of Minkowski distances. To test the effect of the parameter, we varied the order from 1 to 6 and compared the average size of the approximate MEUS's returned by our greedy method for all values in each attribute in the Pima Indians Diabetes data set. In Figure 9, we measure the relative size of a MEUS of value v by dividing the number of tuples in the MEUS by the number of tuples in the projected database of v. Limited by space, the figure shows the results on only three attributes, while the trends in other attributes are consistent.

The average MEUS size decreases when the order increases, though the changes are quite moderate. With a higher order, a sample is penalized more if it has more value pairs of correlations different from the whole data set. On the other hand, small samples with less value pairs in the sample may avoid some penalties, and thus are preferred by a higher order.

Interestingly, the frequent disguise values are stable with respect to the order. Among all the attributes in the Pima Indians Diabetes data set, when the order varies from 1 to 6, only the suspicious disguise value on attribute *Body mass index* changes. When the order is 3 or higher, value 312 replaces value 0 as the top frequent disguise value, but value 0 still has the second highest score among all the values. Therefore, our method is insensitive to the order.

5.4 DV-Score Gain-based versus Contribution Score-based search

In Section 4.3.1, we introduce contribution scores to improve the efficiency. We tested the effectiveness of the contribution scores. That is, we compared the MEUS's computed using the DV-score gain-based greedy search and those computed using the contribution score-based greedy search.

We used the Pima Indians Diabets data set. The results are shown in Table 2. We compared the most frequent disguise values returned by the two search methods on the 7 attributes (i.e., except for the attribute *diabetes pedigree function* where attribute values are of very low frequency as analyzed before). In the table, the most frequent disguise values and the numbers of tuples in the MEUS's returned by the two search methods are shown. We also list the num-

[3]http://www.ics.uci.edu/ mlearn/MLRepository.html.

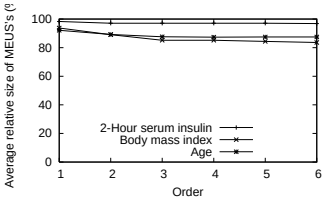

Figure 9: Average relative MEUS size with respect to order in sample quality score.

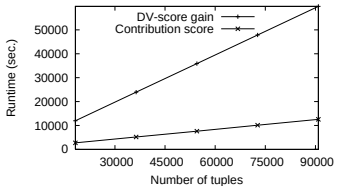

Figure 10: Scalability with respect to number of tuples.

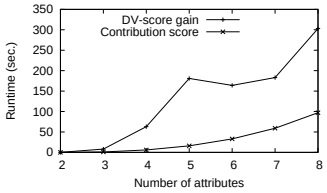

Figure 11: Scalability with respect to dimensionality.

Attribute	DV-score gain		Contribution score		Number of common tuples
	Most frequent disguise value	Number of tuples in the MEUS	Most frequent disguise value	Number of tuples in the MEUS	
Number of times pregnant	0	110	0	110	110
Plasma glucose concentration	91	9	91	9	9
Diastolic blood pressure	70	53	0	35	0
Triceps skin fold thickness	0	220	0	227	217
2-Hour serum insulin	0	374	0	374	374
Body mass index	0	11	0	11	11
Age	21	57	21	62	55

Table 2: The consistency between the DV-score gain-based search and the contribution score-based search.

ber of common tuples in the MEUS's returned by the two methods. We observe the following.

First, in most cases, the two search methods return the most frequent disguise values consistently. In this experiment, the only exception is the attribute *diastolic blood pressure*, where the DV-score gain-based method returns value 70 (the most frequent value on this attribute) and the contribution score method returns value 0 (the suspicious value). It happens the contribution score method picks the correct answer in this case. Overall, the two methods are consistent.

Second, the MEUS's returned by the two methods are highly consistent as long as they pick the same suspicious disguise value. Occasionally, the contribution score method returns a larger MEUS (e.g., for attributes *triceps skin fold thickness* and *age*). The similarity between the MEUS's is at least 86% in our experiments[4].

The experiments clearly show that the contribution score method is consistent with the DV-score gain method. As we can see in the next section, the contribution score method can be much faster and more scalable than the DV-score gain method, and is practical for large real data sets.

5.5 Efficiency and Scalability

We also evaluated the efficiency and the scalability of our approach with respect to the size of the data set and the dimensionality. We tested both the method using the DV-score gain and the one using the contribution score in the greedy search. To obtain large data sets, we duplicate the AERS data set up to 5 times (with 7 attributes). As shown in Figure 10, both methods have a linear scalability, but the contribution score-based search is much more efficient.

We tested the scalability of the two methods with respect to the dimensionality of the data sets using the Pima Indians Diabetes data set. The results are shown in Figure 11. The contribution score-based search is faster. However, the dimensionality curse appears. The runtime of both methods increases exponentially as the number of attributes goes up. The reason is that, with more attributes, there is an exponential increase in the number of value pairs that need to be checked in computing the scores.

[4]The similarity between two sets A and B is defined as $\frac{|A \cap B|}{|A \cup B|}$.

6. CONCLUSIONS

Data cleaning is a foremost step for many data analysis tasks. In this paper, we tackle the important, practical and challenging problem of cleaning disguised missing data. We identify the interesting and useful embedded unbiased sample property of disguised missing data in practice, and propose a novel and practical heuristic approach. Our extensive empirical evaluation using both real data sets and synthetic data sets clearly show that our method is effective and efficient for cleaning large data sets, and can be used in practical applications.

7. REFERENCES

[1] D. DesJardins. Outliers, inliers, and just plain liars – new graphical EDA+ (EDA Plus) techniques for understanding data. In *Proc. SAS User's Group International Conference (SUGI26)*, Long Beach, CA, 2001.

[2] B. Kégl and L. Wang. Boosting on manifolds: Adaptive regularization of base classifiers. In *NIPS'05*.

[3] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley, New York, 1987.

[4] R. Pearson. Mining imperfect data: Dealing with contamination and incomplete records. In *SIAM DM'05*.

[5] R. K. Pearson. The problem of distuised missing data. *ACM SIGKDD Explorations*, 2006.

[6] G. Webb. Further experimental evidence against the utility of occam's razor. *The Journal of Artificial Intelligence Research*, 4:397–417, 1996.