DiMaC: A Disguised Missing Data Cleaning Tool*

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

Jian Pei
School of Computing Science
Simon Fraser University
Burnaby, BC, 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. The very limited previous studies on cleaning disguised missing data highly rely on domain background knowledge in specific applications and may not work well for the cases where the disguise values are inliers.

Recently, we have studied the problem of cleaning disguised missing data systematically, and proposed an effective heuristic approach [2]. In this paper, we present a demonstration of *DiMaC*, a Disguised Missing Data Cleaning tool which can find the frequently used disguise values in data sets without any domain background knowledge. In this demo, we will show (1) the critical techniques of finding suspicious disguise values; (2) the architecture and user interface of *DiMaC* system; (3) an empirical case study on both real and synthetic data sets, which verifies the effectiveness and the efficiency of the techniques; and (4) some challenges arising from real applications and several direction for future work.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation

Keywords

Data Quality, Data Cleaning, Disguised Missing Data

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* [8].

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 values 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, and may impair the quality of data analysis severely. In this demonstration, we will analyze two real data sets, where disguised missing data are detected. Some simple statistics may shift to some anomalous values due to disguised missing data. 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, since 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.

Recently, we have studied the problem of cleaning disguised missing data systematically and made the following contributions [2]. 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. Based on this property, we propose a general framework to identify suspicious frequently used disguise values. Second, mining frequently

*This work was supported in part by an NSERC Discovery grant and an IBM Faculty Award. 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'08, August 24–27, 2008, Las Vegas, Nevada, USA.
ACM 978-1-60558-193-4/08/08.

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.

Based on the above techniques, we developed DiMaC (for Disguised Missing Data Cleaner), a system that can find the possible values frequently used as disguises. In Section 2, we describe disguised missing data and discuss the critical techniques used in *DiMaC*. In Section 3, we propose a system demonstration plan.

2. SYSTEM OVERVIEW

In this section, we describe disguised missing data and present the embedded unbiased sample heuristic of frequent disguise values. We also propose a framework for cleaning disguised missing data, and analyze the computational challenges.

2.1 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.* 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, 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*. 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*.

In data cleaning, we are given the recorded table $\widetilde{T}$, 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.

There are many previous studies on data mining and data analysis with explicitly missing data [5, 4, 7]. However, cleaning disguised missing data is more challenging. Although this problem has been tackled from some angles, the existing approaches often rely on domain knowledge heavily, and are developed for specific applications [6, 1].

2.2 Embedded Unbiased Sample Heuristic

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 [5], 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.

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 $\widetilde{T}_v$.

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.* ■

2.3 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.

Based on the above discussion, we propose a two-phase general framework for cleaning disguised missing data. In the first 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 second phase, those candidates can be verified by domain experts or other data cleaning methods.

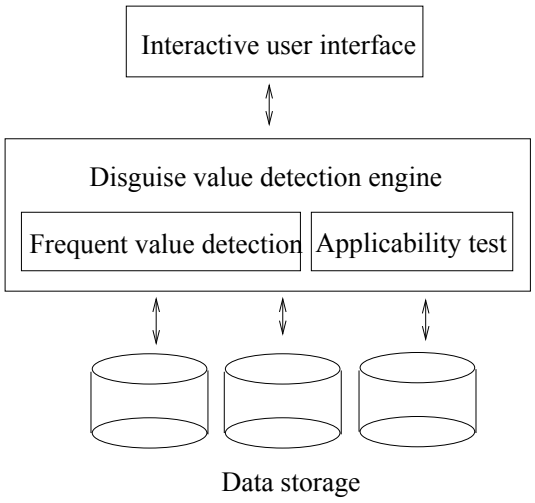


Figure 1: The architecture of DiMac.

DiMaC focuses on the first phase of the framework, which reduces the number of candidates substantially and thus makes the domain experts' analysis effective. The architecture of DiMaC is shown in Figure 1. The *disguise value detection engine* finds frequent attribute values on applicable attributes, and then detects disguise values. Some screenshots of DiMaC are shown in Figure 2.

2.4 Critical Techniques

There are some important technical challenges.

First of all, how can we measure whether a set of tuples is an unbiased sample of a table? We propose *DV-score*, a correlation-based sample quality score, to measure the quality of a sample. By intuition, correlations can capture the distribution of a data set nicely. Thus, the maximal embedded unbiased sample M_v is a subset of the projected database $\widetilde{T}_v$ maximizing the DV-score.

However, the DV-score is not monotonic with respect to the set containment relation, which makes computing the maximal embedded unbiased samples computationally challenging. How can we compute a maximal embedded unbiased sample M_v from the projected database $\widetilde{T}_v$ efficiently?

To tackle the problem practically, we adopt a greedy approach. We start with the projected database of an attribute $\widetilde{T}_v$ as the initial sample. In each iteration, 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.

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. To address the challenge, we develop some techniques to improve the efficiency of the greedy search.

First, since the DV-score of a tuple changes whenever the sample changes, and computing the DV-score gain can be costly, we use a *contribution score* to guide the greedy search. Comparing to computing DV-score gains, contribution scores are often much easier to maintain. Second, we propose an efficient pruning technique to avoid checking each frequent value in an attribute in the greedy algorithm.

Equipped by the above techniques, detecting disguise values is efficient in large data sets. The details of the techniques can be found in [2].

3. DEMONSTRATION PLAN

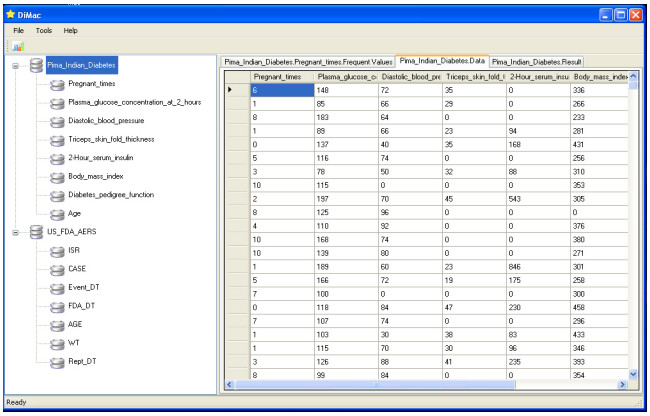
The design and development of DiMaC involve a few challenging database and data cleaning issues, including efficient mining of frequent attribute values and testing the quality of large samples.

We will present our prototype system thoroughly in our demo. Particularly, we will focus on the following aspects.

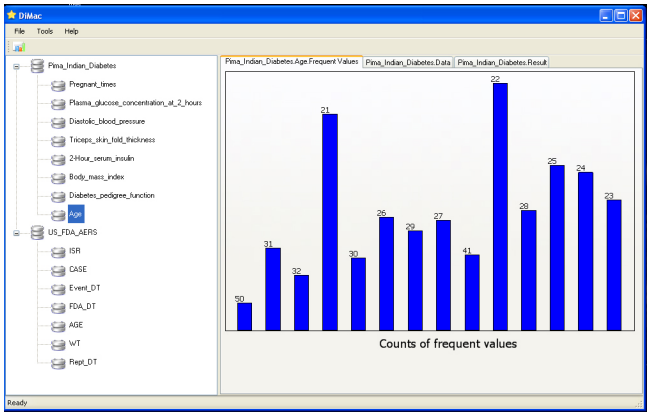
- First, we will *illustrate the sources of disguised missing data and their consequences in data analysis.* We will show the audience different sources of disguised missing data, and why the existing methods cannot handle disguised missing data well. The audience will gain more understanding about the essence of disguised missing data, and raise awareness of disguised missing data in their research work.

- Second, we will *present the technical details in DiMaC, including the efficient implementation.* We will analyze the rationale behind the design, as well as the consideration regarding the scalability issues. This will show the audience how our system can efficiently and effectively identify suspicious frequently used disguise values.

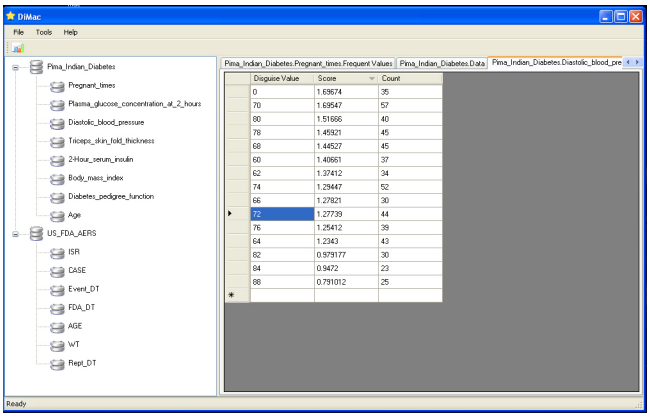
- Third, we will *demonstrate a set of real case studies on our prototype system, and report the experimental results on real and synthetical data sets.* We will demonstrate the disguised missing data detected from two real data sets: the Pima Indians Diabetes database containing records about Pima Indian females who are at least 21 years old and tested either positive or negative for diabetes, and the Adverse Event Reporting



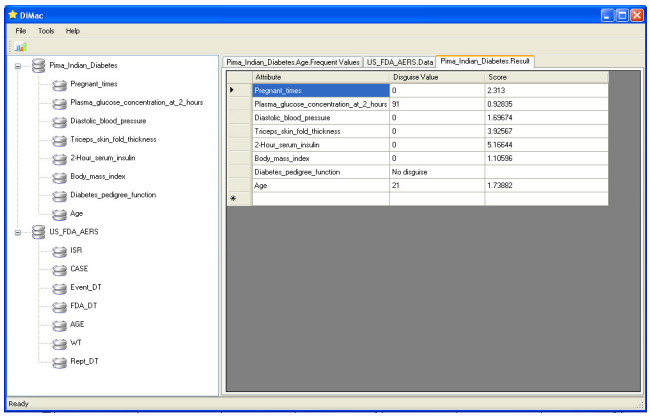
(a) Displaying data sets.



(b) Displaying statistics of attribute values.



(c) Displaying contribution scores of values for one attribute.



(d) Displaying the possible disguise for each attribute.

Figure 2: The screenshots of DiMac.

System data set from the U.S. Food and Drug Administration in the first quarter of 2004. Interestingly, many previous machine learning studies (e.g., [3, 9]) use the Pima Indians Diabetes database but presume that the data set has no missing data. Moreover, we will show some challenging cases that even DiMaC cannot handle well, and the possible extensions. We will share with the audience our interesting findings in the real cases.

- Finally, we will *showcase our prototype system, including a disguise value detection engine and an interactive user interface*. The audience is encouraged to test drive our prototype system on various data sets to further understand disguise missing data and experience the DiMaC prototype system.

4. 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] M. Hua and J. Pei. Cleaning disguised missing data: a heuristic approach. In *KDD*, pages 950–958, 2007.

[3] B. Kégl and L. Wang. Boosting on manifolds: Adaptive regularization of base classifiers. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 665–672, Cambridge, MA, 2005. MIT Press.

[4] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

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

[6] R. Pearson. Mining imperfect data: Dealing with contamination and incomplete records. In *Proc. 2005 SIAM Int. Conf. Data Mining*, New Port Beach, CA, April 2005.

[7] R. K. Pearson. Data mining in the face of contaminated and incomplete records. In *Proc. 2002 SIAM Int. Conf. Data Mining*, Arlington, VA, April 2002.

[8] R. K. Pearson. The problem of distuised missing data. *ACM SIGKDD Explorations*, 8:(1) 83–92, 2006.

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