Information Sciences 465 (2018) 305–322

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Relational data imputation with quality guarantee

Fengfeng Fan[a,b], Zhanhuai Li[a,b], Qun Chen[a,b,*], Lei Chen[c]

[a] School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, 710072, PR China
[b] Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University, MIIT of China, PR China
[c] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, PR China

ARTICLE INFO

*Article history:*
Received 17 May 2017
Revised 19 June 2018
Accepted 8 July 2018
Available online 10 July 2018

*Keywords:*
Missing data imputation
Data cleaning
Quality guarantee
Generalized feature dependency

ABSTRACT

Missing attribute values are prevalent in real relational data, especially the data extracted from the Web. Their accurate imputation is important for ensuring high quality of data analytics. Even though many techniques have been proposed for this task, none of them provides a flexible mechanism for quality control. The lack of quality guarantee may result in many missing data being filled with wrong values, which can easily result in biased data analysis. In this paper, we first propose a novel probabilistic framework based on the concept of Generalized Feature Dependency (GFD). By exploiting the monotonicity between imputation precision and match probability, it enables a flexible mechanism for quality control. We then present the imputation model with precision guarantee and the techniques to maximize recall while meeting a user-specified precision requirement. Finally, we evaluate the performance of the proposed approach on real data. Our extensive experiments show that it has performance advantage over the state-of-the-art alternatives and most importantly, its quality control mechanism is effective.

## 1. Introduction

Missing attribute values are prevalent in real relational data, due to incomplete entry, inaccurate extraction or corrupted sources [1]. Since the missing data may severely impair the integrity of analytical results, it is usually necessary to impute those missing values before delving into data analysis. The existing techniques for relational data imputation can be broadly classified into statistical [14,22,28], nearest neighbor (NN) [2,11,31,40] and machine learning (ML) [12,13,20,26,29] approaches. The statistical approaches seek to fill the missing data such that statistical properties of interest (e.g., mean or variance) are maximally maintained. The NN-based approaches fill a record's missing attribute value based on its neighbors. In comparison, the ML-based approaches first train a model over complete records, and then impute the missing values by either classification or regression. It is noteworthy that the NN-based approaches are usually effective in the cases where there exist a lot of duplicate values or highly similar records whereas in the cases where only a limited number of duplicate values are available, it has been empirically shown that the ML-based approaches usually perform better.

Unfortunately, none of the existing solutions provides a mechanism for quality control. The lack of quality guarantee may result in many missing data being filled with wrong values, which can easily result in biased data analysis. Therefore, it is often desirable to leave a missing value intact rather than fill it with a wrong one. To this end, we propose a new probabilistic framework in this paper. We motivate our approach by comparing it with the NN-based approach using the

* Corresponding author.
*E-mail address:* chenbenben@nwpu.edu.cn (Q. Chen).

https://doi.org/10.1016/j.ins.2018.07.017
0020-0255/© 2018 Elsevier Inc. All rights reserved.

**Table 1**
A running example.

| ID | Author | Title | Journal |
|---|---|---|---|
| $t_1$ | Mecca, G. and Bonner, A. J. | **Query** languages for sequence **databases**: termination and complexity | IEEE Transactions on Knowledge & Data Engineering |
| $t_2$ | Chen, Ming Syan and Han, Jiawei and Yu, P. S. | Data Mining: An Overview from a **Database** Perspective | IEEE Transactions on Knowledge & Data Engineering |
| $t_3$ | Smyth, P. and Goodman, R. M. | An Information Theoretic Approach to Rule Induction from **Databases** | IEEE Transactions on Knowledge & Data Engineering |
| $t_4$ | Lam, Kam Yiu and Kuo, Tei Wei and Kao, Ben and Lee, Tony S. H and Cheng, Reynold | Evaluation of Concurrency Control Strategies for Mixed Soft Real-Time **Database** Systems | Information Systems |
| $t_5$ | Ngu, Anne H. H. and Sheng, Quan Z. and Du, Q. Huynh and Lei, Ron | Combining multi-visual features for efficient indexing in a large image **database** | VLDB Journal |
| $t_6$ | Gao, Yunjun and Zheng, Baihua and Chen, Gencai and Li, Qing | Optimal-Location-Selection **Query** Processing in Spatial **Databases** | _ |

example shown in Table 1. Suppose that some missing values for the attribute Journal need to be imputed, a typical NN approach reasons about a missing Journal value using the similarity rule (SR) defined as follows:

$$SR_1 : [\text{Author}, \text{Title}] \rightarrow [\text{Journal}]$$

$SR_1$ states that if two records have similar Author and Title values, their Journal values should also be similar. According to $SR_1$, a record's missing Journal value can be filled with the Journal value of its nearest neighbor measured on Author and Title. It can be observed that the major challenge of the NN-based approach is to answer the question of how similar is similar. If the similarity threshold is set high, it can easily suffer from neighbor sparsity. A low threshold is instead prone to incorrect imputation.

In contrast to the NN-based approach, we propose to take a probabilistic perspective. It first extracts features from the existing attribute values and then probabilistically reasons about a missing value by analyzing its correlation with the extracted features. In the example shown in Table 1, it can be observed that the journal of "IEEE Transactions on Knowledge and Data Engineering (TKDE)" is correlated with three quite different paper titles in the records of $t_1$, $t_2$ and $t_3$. The NN approach thus fails to detect the correlation existing between the paper titles and the journal venues. However, it can be observed that the three paper titles contain a common keyword "database". It follows that the keyword "database" is probably correlated with the "TKDE" journal. The proposed probabilistic approach represents the title keywords as the discriminating features of journals, and quantifies their correlation by a new form of functional dependency between features and attribute values, called the Generalized Feature Dependency (GFD). In a GFD, the left-hand side (LHS) represents a feature while the right-hand side (RHS) represents the probability distribution of its correlated attribute values. For instance, in the example shown in Table 1, by a GFD, the presence of the feature "database" at the Title attribute indicates that the probability of its Journal attribute value being TKDE is 60%, while the probability of being "VLDB Journal" or "Information Systems" is both 20%.

The most attractive property of the proposed probabilistic approach is that it enables a mechanism for quality control. By taking advantage of the statistical monotonicity relationship between imputation precision and match probability, it can flexibly enforce precision guarantee. Specifically, given a user-specified precision level, it first learns an optimal threshold of match probability based on non-missing values, and then imputes a missing value if and only if its estimated match probability exceeds the learned threshold. The major contributions of this paper can be summarized as follows:

1. We propose a novel probabilistic framework for relational data imputation based on the concept of generalized feature dependency.
2. We propose an imputation model with precision guarantee. We present the techniques to maximize recall while ensuring a user-specified precision level.
3. We demonstrate by extensive experiments on real data that the proposed framework performs better than the state-of-the-art alternatives and most importantly, its quality control mechanism is effective.

For the sake of presentation simplicity, we summarize the frequently used notations in Table 2. The rest of this paper is organized as follows: Section 2 defines the imputation task; Section 3 presents the probabilistic framework; Section 4 describes the imputation model with precision guarantee; Section 5 presents the algorithms and analyzes their complexities; Section 6 presents the empirical evaluation results; Section 7 reviews more related work, and finally Section 8 concludes the paper.

## 2. Task statement

Given a tuple $r_i$ in a relational table $R$, its value at the attribute $B$ is deemed to be missing if it is equal to null. The goal of data imputation is to fill in a value of $B$ for $r_i$ as accurately as possible based on the available information. In general, the

Table 2
Notations.

| Symbol | Notation |
|---|---|
| $R$ | a relational table; |
| $r_i$ | a tuple in $R$; |
| $f_i$ | a feature of the record entities in $R$; |
| $A$, $B$ | the attributes of $R$; |
| $a_i$ ($b_i$) | an attribute value at $A$ ($B$); |
| $A(r_i)$ | the attribute value of $r_i$ at $A$; |
| $X_i$ | the set of features extracted from $r_i$; |
| $Dom(A)$ | the value domain of the attribute $A$; |
| $R_I$ | the set of tuples whose values at the imputation attribute are missing; |
| $R_C$ | the set of tuples whose values at the imputation attribute are *not* missing; |
| $R_W$ | a subset of $R_C$, which is used to estimate match probability; |
| $R_V$ | a subset of $R_C$, which is used to learn optimal imputation model. |

Table 3
An example of imputation with quality guarantee.

| $r_i$ | $P(b_1\|X_i)$ | $P(b_2\|X_i)$ | $P(b^*\|X_i)$ | $b^*$ | $b'$ | Correct | Accepted |
|---|---|---|---|---|---|---|---|
| $r_1$ | 1.0 | 0.0 | 1.0 | $b_1$ | $b_1$ | Y | Y |
| $r_2$ | 0.1 | 0.9 | 0.9 | $b_2$ | $b_2$ | Y | Y |
| $r_3$ | 0.51 | 0.49 | 0.51 | $b_1$ | $b_2$ | N | N |

available information consists of all the data in $R$. It can be observed that with regard to the imputation at $r_i$, the non-null attribute values of $r_i$ are of special interest. We represent the available information of $r_i$ by its feature set, which is denoted by $X_i$. Optimally, for the imputation of $B$ at $r_i$, the imputed value $b_j$ should have the maximal conditional probability, $P(b_j|X_i)$. Therefore, we formally define the probabilistic imputation task without quality guarantee as follows:

**Definition 1** (Traditional Relational Data Imputation)**.** Given a relational table $R$ with missing values at the attribute $B$, the imputation task is to build a probabilistic model $\mathcal{M}$: for each $r_i \in R_I$, $\mathcal{M}$ chooses a candidate, $b_k \in Dom(B)$, for the missing value, such that $b_k = \underset{j}{argmax}(P_{\mathcal{M}}(b_j|X_i))$, in which $X_i$ denotes the feature set extracted from $r_i$ and $P_{\mathcal{M}}(b_j|X_i))$ denotes the conditional probability estimated by $\mathcal{M}$.

It is noteworthy that the task specified in Definition 1 does not enforce any quality guarantee on the resulting imputations. It is clear that even though an imputed value $b_k$ achieves the maximal conditional probability, its probability may be so low that $b_k$ has only a small chance to be the true value. In this case, it is usually undesirable to fill in the missing value with $b_k$. A more desirable practice is instead to impute a missing value with a candidate *if and only if the candidate has a high probability to be the true value*.

In this paper, we enforce quality guarantee by ensuring the precision level, which denotes the proportion of the correct imputations in all the executed imputations. The task of relational data imputation with quality guarantee is therefore to build a probabilistic model that can maximize the recall while ensuring a user-specified precision level with a given confidence. Note that provided with a precision requirement, maximizing recall corresponds to maximizing the number of filled missing values. We formally define the task by a constrained optimization problem as follows:

**Definition 2** (Relational Data Imputation with Quality Guarantee)**.** Given a relational table $R$ with missing values at the attribute $B$, a precision level $\phi$ and a confidence level $\gamma$, the probabilistic imputation task with quality guarantee is defined as follows:

$$\begin{aligned} &\mathcal{M}^* = \underset{\mathcal{M}}{argmax}(|R_{\mathcal{M}}|) \\ &s.t. \quad P(prec(\mathcal{M}, R) \geq \phi) \geq \gamma, \end{aligned}$$

in which $\mathcal{M}$ denotes a probabilistic model, $\mathcal{M}^*$ denotes the optimal model that maximizes the number of filled missing values while ensuring the precision level of $\phi$, $R_{\mathcal{M}}$ denotes the set of tuples whose missing values at $B$ are filled by $\mathcal{M}$, $|R_{\mathcal{M}}|$ denotes the number of tuples in $R_{\mathcal{M}}$, and $prec(\mathcal{M}, R)$ denotes the achieved imputation precision of $\mathcal{M}$ on $R$.

We illustrate the difference between the traditional imputation and the imputation with quality guarantee by the following toy example:

**Example 1.** Suppose that $R$ contains three tuples $r_1$, $r_2$, $r_3 \in R_I$ with the missing values at the attribute $B$, and there are two candidate values $b_1$, $b_2 \in Dom(B)$. Let $b^*$ and $b'$ denote the candidate with the maximal posterior probability and the ground truth respectively. As shown in Table 3, the imputations for $r_1[B]$ and $r_2[B]$ should be accepted due to the high match probabilities of $P(b^*|X_1)$ and $P(b^*|X_2)$, while the imputation for $r3[B]$ should be rejected since $P(b^*|X_3) = 0.51$ implies a great risk of incorrect imputation ($b^* \neq b'$). Comparatively, the traditional imputation will accept all the candidates suggested by a probabilistic model, while the imputation with quality guarantee would only accept the candidates with sufficiently high posterior probabilities.

## 3. Probabilistic framework

In this section, we propose a probabilistic framework, which computes the match probability of a candidate (i.e., its probability of being the true value) by quantifying its correlation with features. We first define the concept of GFD (Generalized Feature Dependency), then describe the process of feature extraction and finally present the techniques for match probability estimation.

### 3.1. Generalized feature dependency

Before defining GFD, we first define feature dependency, which specifies the relationship between a feature and an attribute value, as follows:

**Definition 3.** Given a feature, $f_i$, of the attribute A, and an attribute value, $b_j$, of $B$ in $R$, $f_i$ and $b_j$ satisfy the relationship of feature dependency, denoted by $FD$: $f_i \rightarrow b_j$, iff $\forall\ r_k \in R$, if $r_k$ contains the feature $f_i$ at $A$, then $B(r_k) = b_j$.

The feature dependency $FD$: $f_i \rightarrow b_j$ means that the feature $f_i$ can uniquely determine the value at $B$. For instance, in the example of Table 1, assume that if a paper contains the keyword “query” in its title, its Journal attribute value should be “TKDE”. Then, a feature dependency exists between the feature “query” and the attribute value “TKDE”. Obviously, a feature dependency can be highly effective in reasoning about missing values. In Table 1, if a tuple has a missing Journal value and its title contains the keyword “query”, it can be reasoned by the feature dependency that the missing Journal value should be “TKDE”.

Unfortunately, the validity of feature dependency depends on the uniqueness of attribute values on the right hand side. It can be observed that for representing the correlation between features and attribute values, feature dependency is usually too rigid. In Table 1, it can be expected that there exist very few features of title keyword or author name that can determine the value at the Journal attribute. As a result, very few missing values at Journal could be filled by feature dependencies. To enable more flexible representation of the correlation between features and attribute values, we introduce the concept of generalized feature dependency. As a generalization of feature dependency, GFD replaces a single value with a multi-value probability distribution at its right-hand side. We formally define GFD as follows:

**Definition 4.** Suppose that $f_i$ denotes a feature of $A$ and $S_b$ denotes a set of attribute values of $B$ in $R$. The generalized feature dependency between $f_i$ and $S_b$ is represented by

$$GFD : f_i \xrightarrow{\mathcal{D}(P(b_j|f_i))} S_b,$$

in which $\forall b_j \in S_b$, $P(b_j|f_i)$ denotes the probability of a tuple's attribute value at $B$ being $b_j$ given the presence of the feature $f_i$ at $A$, $\mathcal{D}(P(b_j|f_i))$ denotes the distribution of $P(b_j|f_i)$ with regard to all the distinct $B$ values in $S_b$, subject to $\Sigma_j P(b_j|f_i)=1$.

In the special case that the feature $f_i$ can determine a unique attribute value at $B$, which is supposed to be $b^*$, we have

$$P(b_j|f_i) = \begin{cases} 1 & \text{if } b_j = b^*, \\ 0 & \text{otherwise}; \end{cases} \tag{1}$$

Then, generalized feature dependency degenerates into feature dependency. It is also interesting to observe that if each attribute value of $A$ corresponds to a feature and each feature can uniquely determine the attribute value of $B$, the feature dependencies between the attribute values at $A$ and $B$ constitute a traditional functional dependency between $A$ and $B$.

### 3.2. Feature extraction

Obviously, each attribute value can be considered as a feature. For instance, in Table 1, each paper title corresponds to a feature. However, the effectiveness of a GFD for missing value imputation depends on its feature's repeated occurrences in multiple tuples. The features of attribute values would have limited prediction power if they have few redundancies in a dataset. In Table 1, if each tuple corresponds to a unique paper and each paper has a distinct title, then the feature dependencies between Title values and Journal values are powerless with regard to reasoning about the missing Journal values. Therefore, for string attribute values, we extract $n$-grams, which are widely used in computational linguistics [5], as features. In Table 1, $n$-grams can be title keywords (e.g. “database” and “database systems”) or author names (e.g., “Yu, P.S”). For numerical attribute values, the entire values should be treated as features.

### 3.3. Probability estimation

Observing that a tuple in $R$ usually contains multiple features, we first present the methods to compute the conditional probability, $P(b_j|f_i)$, for a single feature, and then describe how to compute the unified probability of $P(b_j|X_i)$ by aggregating $P(b_j|f_i)$, in which $X_i$ contains multiple features.

### 3.3.1. Computing P(b_j|f_i)

We estimate the conditional probabilities based on the tuples in $R_W$, whose attribute values at $B$ are not missing. If the latter are discrete, $P(b_j|f_i)$ can be simply computed by

$$P(b_j|f_i) = \frac{freq(b_j, f_i) + \epsilon}{freq(f_i) + N_B \cdot \epsilon}, \tag{2}$$

where $freq(f_i)$ denotes the total number of tuples containing the feature $f_i$, $freq(b_j, f_i)$ denotes the total number of tuples containing both $f_i$ and $b_j$, $N_B$ denotes the cardinality of $Dom(B)$ and $\epsilon$ is a smoothing parameter.

If the attribute values at $B$ are continuous, we resort to the technique of *kernel density estimation* [9] to approximate $P(b_j|f_i)$. Suppose that there are $m$ tuples containing the feature $f_i$ in $R_W$. The conditional probability is estimated by

$$P(b_j|f_i) = \frac{1}{m}\sum_{k=1}^{m} K\left(\frac{b_j - b_k}{\mu}\right), \tag{3}$$

where $b_k$ represent a tuple's $B$ value, $K(\cdot)$ represents a kernel function, and $\mu$ is a smoothing bandwidth parameter. In practice, the Gaussian function is usually used as kernel.

### 3.3.2. Computing P(b_j|X_i)

Formally, the framework measures the unified probability of a candidate $b_j$, denoted by $U(b_j|X_i)$, by a linear model as follows:

$$U(b_j|X_i) = \sum_{\forall f_k \in X_i} w_k \cdot P(b_j|f_k), \tag{4}$$

where $f_k$ denotes a feature in $X_i$, $w_k$ denotes the feature weight. In Eq. (4), the feature weight $w_k$ indicates a feature's power in determining the attribute value at $B$. We measure the power of a feature by

$$pow(f_i, B) = \sum_{\forall b_j \in Dom(B)} P(b_j|f_i)^{\lambda}, \tag{5}$$

in which $\lambda \geq 1$ is a tuning parameter. In the case of $\lambda = 1$, all the features have the same prediction power of 1. If $\lambda > 1$, the prediction power of a feature would increase with the non-uniformity of its probability distribution. For instance, with $\lambda = 2$, $pow(f_k, B)$ will take the minimal value of $\frac{1}{N}$ given a uniform distribution over $B$ values, in which $N$ denotes the number of distinct $B$ values. In comparison, if $f_i$ can instead determine a unique value for $B$, $pow(f_k, B)$ will take the maximal value of 1. It can also be observed that if $\lambda \gg 1$, $P(b_j|f_k)$ with the maximal value would dominate a feature's prediction power. That is, increasing the value of $\lambda$ would enlarge the relative prediction power for the feature with a more skewed probability distribution.

**Example 2.** In the example shown in Table 1, by GFD, the feature "database" of the attribute Title indicates that the probability of the corresponding value at Journal being "TKDE" is 60%, its probability of being "VLDB Journal" or "Information Systems" is both 20%. With $\lambda = 2$, $pow(\text{"database"}, \text{Journal}) = \frac{11}{25} = 0.44$.

With $w_k = pow(f_k, B)$, the unified probability of $b_j$ can be represented by

$$U(b_j|X_i) = \sum_{\forall f_k \in X_i} pow(f_k, B) \cdot P(b_j|f_k) \tag{6}$$

Due to length variety of attribute values, the number of extracted features may vary significantly in different tuples. Therefore, we use the **softmax** transformation [33], which has been widely used in machine learning, to normalize the unified probabilities into posterior probabilities at a uniform scale as follows:

$$P(b_j|X_i) = \frac{\exp(\tau \cdot U(b_j|X_i))}{\sum_{k=1}^{N_B} \exp(\tau \cdot U(b_k|X_i))} \tag{7}$$

where $exp(\cdot)$ denotes the natural exponential function, $N_B$ denotes the cardinality of $Dom(B)$ and $\tau > 0$ is a tuning parameter. Intuitively, the posterior probability $P(b_j|X_i)$ represents the relative preference of $b_j$ over other candidates for the imputation in $r_i$. In the extreme case of $\tau = 0$, all the posterior probabilities $P(b_j|X_i)$ hold equal values. With $\tau > 0$, the candidate with the largest unified probability has the correspondingly largest posterior probability. It can be observed that increasing the value of $\tau$ would enlarge the relative preference for the candidate with the largest unified probability over other candidates. In the extreme case of $\tau \to \infty$, the candidate with the largest unified probability (e.g., $b^*$) would have the posterior probability close to 1, or $P(b^*|X_i) \approx 1$, and the posterior probabilities of other candidates would be close to 0, or $P(b_j|X_i) \approx 0$ if $b_j \neq b^*$.

It can be observed that in the framework without quality guarantee, the parameter $\tau$ can not affect imputation results: it is therefore set to be the default value of 1. For quality control, the values of $\lambda$ and $\tau$ in Eqs. (5) and (7) can be tuned for maximizing recall. Their tuning techniques would be presented in details in Section 4.

**Table 4**
An example of GFD matrix.

| Journal/Feature | "Query" | "Database" |
|---|---|---|
| TKDE | 1.0 | 0.6 |
| VLDB Journal | 0 | 0.2 |
| Information Systems | 0 | 0.2 |

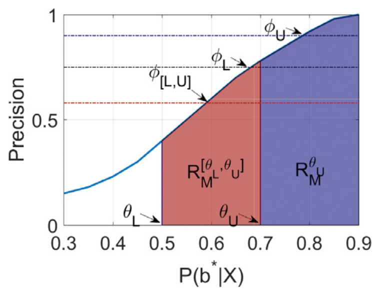


**Fig. 1.** Precision monotonicity.

**Example 3.** Suppose that in Table 1, $f_1$ = "Query" and $f_2$ = "Database" are two features of the Title attribute. The GFD matrix is shown in Table 4. With $\lambda = 2$, the feature weights can be computed by $pow(f_1, B) = 1$, and $pow(f_2, B) = 0.44$. We denote the Journal attribute values, "TKDE", "VLDB Journal" and "Information Systems" by $b_1$, $b_2$ and $b_3$ respectively. For the missing value imputation in $t_6$, the unified probabilities of the three candidates can be computed by $U(b_1|X_6) = 1.0 * 1 + 0.44 * 0.6 = 1.264$, $U(b_2|X_6) = 1.0 * 0 + 0.44 * 0.2 = 0.088$, and $U(b_3|X_6) = 1.0 * 0 + 0.44 * 0.2 = 0.088$. With $\tau = 2$, their final posterior probabilities can be computed by Eq. (7), as $P(b_1|X_6) = 0.84$, $P(b_2|X_6) = 0.08$ and $P(b_3|X_6) = 0.08$.

## 4. Quality control

As defined in Definition 2, the optimization objective of quality control is to build a probabilistic model that can maximize recall while meeting a user-specified precision level with a given confidence. We represent a model by $\mathcal{M}(\lambda, \tau, \theta)$, in which $\lambda$ and $\tau$ are the parameters defined in Eqs. (5) and (7) respectively, and $\theta$ denotes the match probability threshold. The mechanism of quality control ensures imputation precision by setting an appropriate threshold on match probability. In other words, a filling candidate $b^*$ for the tuple $r_i$ would be accepted by $\mathcal{M}(\lambda, \tau, \theta)$ if and only if its estimated match probability exceeds the threshold, or $P(b^*|X_i) \geq \theta$.

In the rest of this section, we first describe the monotonicity property between match probability threshold and achieved precision level in Section 4.1, then present the techniques for model optimization in Section 4.2, and finally give the theoretical bounds in Section 4.3.

### 4.1. Precision monotonicity

Intuitively, it can be reasoned that the higher the value of $P(b^*|X_i)$ is, the more probably it is the true value. We represent this property by

$$P(b^* = b^r) \sim P(b^*|X_i) \tag{8}$$

where $b^*$ is a filling value and $b^r$ represents the true value. With this assumption, we formally establish the monotonicity property between the match probability threshold and the achieved precision level by the following lemma:

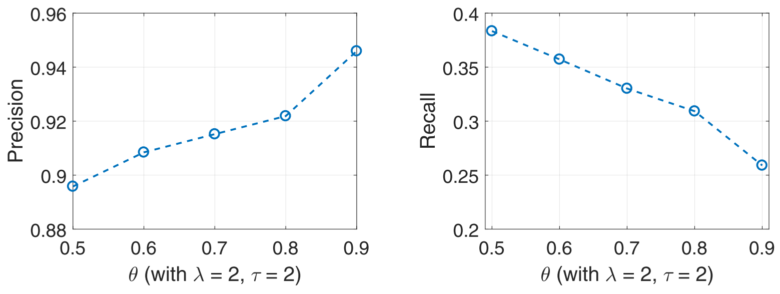
**Lemma 1.** *Given R and two imputation models with the same parameter values for $\lambda$ and $\tau$, $\mathcal{M}(\lambda, \tau, \theta_L)$ and $\mathcal{M}(\lambda, \tau, \theta_U)$, suppose that the imputation model satisfies the monotonicity of precision on R. If $\theta_L \leq \theta_U$, then $\phi_L \leq \phi_U$, where $\phi_i$ denotes the achieved precision level of $\mathcal{M}(\lambda, \tau, \theta_i)$ on R.*

**Proof.** Assuming that $R_{\mathcal{M}}^{\theta_L}$ and $R_{\mathcal{M}}^{\theta_U}$ denote the set of tuples imputed by $\mathcal{M}(\lambda, \tau, \theta_L)$ and $\mathcal{M}(\lambda, \tau, \theta_U)$ respectively, and $R_{\mathcal{M}}^{[\theta_L, \theta_U]}$ denotes the set of tuples with the match probabilities between $\theta_L$ and $\theta_U$, then we have

$$R_{\mathcal{M}}^{\theta_L} = R_{\mathcal{M}}^{\theta_U} \cup R_{\mathcal{M}}^{[\theta_L, \theta_U]}$$

With $\phi_{[L,U]} \leq \phi_U$ as shown in Fig. 1, we have

$$\phi_L = \frac{\phi_U \cdot N_U + \phi_{[L,U]} \cdot N_{[L,U]}}{N_U + N_{[L,U]}} \leq \frac{\phi_U \cdot N_U + \phi_U \cdot N_{[L,U]}}{N_U + N_{[L,U]}} = \phi_U$$



(a) Precision vs. $\theta$ (b) Recall vs. $\theta$

**Fig. 2.** Impact of $\theta$ on precision & recall.

where $\phi_U$, $\phi_L$ and $\phi_{[L,\,U]}$ denote the precisions for $R_{\mathcal{M}}^{\theta_U}$, $R_{\mathcal{M}}^{\theta_L}$ and $R_{\mathcal{M}}^{[\theta_L,\theta_U]}$ respectively, and $N_U$ and $N_{[L,\,U]}$ denoting the cardinalities of $R_{\mathcal{M}}^{\theta_U}$ and $R_{\mathcal{M}}^{[\theta_L,\theta_U]}$ respectively. $\square$

We have also empirically verified the *monotonicity* property between $\theta$ and $\phi$ on the real dataset of CiteSeer[1]. We set $\lambda$=2 and $\tau$=2. The detailed evaluation results are presented in Fig. 2. It can be observed that the achieved precision level increases with the threshold $\theta$, while the achieved recall level decreases with $\theta$.

According to the precision monotonicity, the achieved precision level of an imputation model, $\mathcal{M}(\lambda, \tau, \theta)$, increases with the threshold $\theta$. Therefore, precision level can be enforced by only imputing the candidates whose match probabilities exceed an underlying threshold, or $P(b^*|X_i) \geq \theta$.

## 4.2. Model optimization

To facilitate quality control, we partition $R_C$, the set of tuples whose values at the attribute $B$ are not missing, into two sets, $R_W$ and $R_V$. The set $R_W$ is used to estimate the probability $P(b_j|X_i)$, and the set $R_V$ is instead used for optimizing the model by parameter tuning. The process of model optimization searches an optimal model on the validation set $R_V$, and uses its performance on $R_V$ to predict its performance on $R_I$. With the assumption that $R_V$ and $R_I$ are drawn independently from same underlying distribution, a model's high performance on $R_V$ would result in its similarly high performance on $R_I$.

Note that the achieved recall of a model, $\mathcal{M}(\lambda, \tau, \theta)$, on the validation set $R_V$ decreases with the threshold $\theta$. Therefore, given the parameter values of $\lambda$ and $\tau$, the recall level of a model on $R_V$ can be maximized by minimizing the threshold of $\theta$ while ensuring the precision level of $\phi$. The optimal threshold $\theta$ can be identified by the following two steps:

1. Order all the filling candidates by their match probabilities $P(b^*|X_i)$ in a decreasing order;
2. Iteratively impute a candidate in the ordered list until the precision level falls below $\phi$.

The objective of model optimization is therefore to search for the values of $\lambda$ and $\tau$ for the model, $\mathcal{M}(\lambda, \tau, \theta)$, such that the achieved recall level on $R_V$ is maximized while the achieved precision meets the requirement $\phi$. We formally define the optimization problem as follows:

$$\begin{aligned} &(\lambda^*, \tau^*) = \underset{\mathcal{M}}{argmax}(|R_{\mathcal{M}}|) \\ &s.t. \quad R_{\mathcal{M}} = \text{imputedSet}(\lambda, \tau, \phi) \end{aligned} \tag{9}$$

where $\phi$ denotes the user-specified precision level, $\mathcal{M}$ denotes a model that meets the precision level $\phi$, and the function *imputedSet*($\lambda$, $\tau$, $\phi$) returns the tuple set imputed by the model $\mathcal{M}$ with the parameters of $\lambda$ and $\tau$.

Since the 0–1 loss is employed in assessing the precision of imputations on the fly, the objective function Eq. 9 is neither smooth nor convex. Therefore, neither convex optimization [4,16] nor gradient-descent algorithms [17,27] can be directly applied in optimizing the imputation model. It is noteworthy that in practice, the recommended search spaces for $\lambda$ and $\tau$ are both limited (e.g. $1 \leq \lambda \leq 5$ and $1 \leq \tau \leq 10$). Increasing the value of $\tau$ would cause the threshold of $\theta$ to increase as well. A threshold close to the upper limit of 1 is however undesirable because it would result in many filling candidates failing to meet the threshold. Increasing the value of $\lambda$ would weaken the contribution from low-weight features, thus cause the filling recall to decline. We therefore propose a brute-force solution and a more efficient heuristic alternative, which will be detailed in the Sections 5.2 and 5.3, respectively.

We have also empirically examined the impact of the values of $\lambda$ and $\tau$ on the real dataset of CiteSeer. We set $\phi$ = 0.85, and vary the value of $\lambda$ from 1 to 3 and the value of $\tau$ from 1 to 6. The step sizes of $\lambda$ and $\tau$ are set to 0.3 and 1 respectively. The detailed evaluations are presented in Fig. 3. It can be observed that the objective function is neither smooth nor convex, and both $\lambda$ and $\tau$ can influence the imputations.

[1] https://www.cs.purdue.edu/commugrate/data/citeseer/.

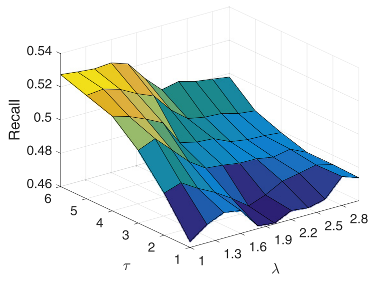


**Fig. 3.** Impact of $q$ and $\tau$ for recall.

### *4.3. Bounds of precision*

The theory of VC dimension[2] is usually used to predict the probabilistic upper bound on the test error of a classification model. Vapnik [35] proved that the probability of the classification error on a test data, which is drawn independently from the same distribution as the training data, can be given by

$$P(\Upsilon_{test} \le \Upsilon_{train} + \Delta) = 1 - \eta, \quad \Delta = \sqrt{\frac{1}{N}[d_{\mathcal{M}}(log(\frac{2N}{d_{\mathcal{M}}}) + 1) - log(\frac{\eta}{4})]} \tag{10}$$

where $\Upsilon_{train}$ and $\Upsilon_{test}$ denote the misclassification rates over the training data and the test data respectively, $\Delta$ denotes their performance gap, $N$ denotes the size of the training data, $d_{\mathcal{M}}$ is the VC dimension (for Vapnik–Chervonenkis dimension) of the classification model $\mathcal{M}$ and $0 < \eta < 1$. Note that VC dimension measures the capacity of a space of functions that can be learned by a statistical classification model. It is defined as the cardinality of the largest set of points that a model can shatter.

In our scenario, the model $\mathcal{M}$ classifies the tuples in $R_I$ into two categories, which consist of the tuples whose missing values would be imputed and the tuples whose missing values would *not* be imputed respectively. $R_V$ and $R_I$ correspond to the training data and the test data respectively. Therefore, the achieved precision of the optimal model, $\mathcal{M}^*$, over $R_I$ can be estimated by

$$P(\phi_I \ge \phi_V - \Delta) = 1 - \eta \tag{11}$$

Given a precision requirement $\phi$ and a confidence level $\gamma$, we can ensure the precision of $\mathcal{M}^*$ by setting $\eta = 1 - \gamma$ and $\phi = \phi_V - \Delta$, in which $\phi_V$ denotes the achieved precision of $\mathcal{M}^*$ over $R_V$. Accordingly, the lower bound of the precision of $\mathcal{M}^*$ over $R_V$ should be $\phi + \Delta$, or $\phi_V \ge \phi + \Delta$. Eq. (11) dictates the lower bound of the precision of $\mathcal{M}^*$ over $R_I$. Note that the performance gap of $\mathcal{M}^*$ over $R_V$ and $R_I$, $\Delta$, decreases with the size of $R_V$. On the VC dimension of our imputation model, we have the following lemma:

**Lemma 2.** *The VC dimension of the classifier corresponding to the model of $\mathcal{M}(\lambda, \tau, \theta)$ is equal to 1.*

**Proof.** Consider the two tuples $r_1$ and $r_2$ having missing values for the attribute $B$ in $R$. Suppose that both of them have only one feature, $f_1$ in $r_1$ and $f_2$ in $r_2$, and each missing value has only two candidates, $b_1$ and $b_2$. Also suppose that the two data points corresponding to $r_1$ and $r_2$ can be represented by the feature dependency vectors of $D_1$=($w_{11}$, $w_{12}$) and $D_2$=($w_{21}$, $w_{22}$) respectively, in which $w_{ij} = P(b_j|f_i)$, $w_{11} + w_{12} = 1$, $w_{21} + w_{22} = 1$ and $w_{11} > w_{21} > 0.5$. Since $w_{11} > w_{21} > 0.5$, it can be observed that $\sigma_{\mathcal{M}}(D_1) \ge \sigma_{\mathcal{M}}(D_2)$, in which $\sigma_{\mathcal{M}}(D_i)$ denotes the estimated probability of the filling candidate for $D_i$ based on the model $\mathcal{M}$. Consider the label assignment of $L(D_1)$="-" and $L(D_2)$="+", in which the label of "+" (or "-") means that the target tuple is imputed (or not imputed) by $\mathcal{M}$. It can be observed that $\mathcal{M}$ cannot shatter $D_1$ and $D_2$ with the label assignment as shown above. Proof completed. □

According to Eq. (10) and Lemma 2, on the theoretical precision bound of the optimal model on $R_I$, we have the following theorem:

**Theorem 1.** *Suppose that $R_V$ and $R_I$ are drawn independently from the same distribution. Given a precision requirement of $\phi$ and a confidence level of $\gamma$, the imputation model $\mathcal{M}$, which achieves the filling precision $\phi_V = \phi + \Delta$ on $R_V$, would achieve the precision of $\phi$ on $R_I$ with the probability of at least $\gamma$, in which*

$$\Delta = \sqrt{\frac{1}{|R_V|}\left[(log(2|R_V|) + 1) - log(\frac{1-\gamma}{4})\right]}$$

[2] https://en.wikipedia.org/wiki/VC_dimension

**Example 4.** Suppose that $R_V$ has 10,000 records. Given the precision requirement of 0.8 and the confidence level of 0.9, the performance gap is 0.038 according to Theorem 1, or $\Delta = 0.038$. Therefore, an imputation model, which achieves the precision of 0.838 on $R_V$, would achieve the desired precision of 0.8 on $R_I$ with the confidence of 0.9.

## 5. Algorithms

In this section, we present the necessary algorithms for identifying the optimal model, $\mathcal{M}(\lambda, \tau, \theta)$ and analyze their time complexity. The imputation algorithm based on $\mathcal{M}(\lambda, \tau, \theta)$ is straightforward, thus omitted here.

### *5.1. Computing the GFD matrix*

For the task of missing value imputation, we represent the generalized feature dependencies by a matrix, **W**, in which each row and column correspond to a feature and an attribute value respectively, and the value of $\mathbf{W}[f_i, b_j]$ represents the conditional probability of $P(b_j|f_i)$. A feature refers to an $n$-gram at any attribute except $B$ in $R$, in which $n$ is usually set to be a small number. In the example shown in Table 1, both author names and keywords in the titles represent features. If $n \leq 2$, then both 1-grams (e.g. "database") and 2-grams (e.g. "database systems") are considered as features. Our implementation in Section 6 used the feature extraction functions provided by the open-source scikit-learn project [25]. It is worthy to point out that the imputation framework does not require to select features for performance optimization: any n-gram in the attribute values can be extracted as a feature. However, not all features are useful for the imputation in $R_I$. We define the concept of effective feature as follows:

**Definition 5.** A feature $f_E$ is *effective* if and only if $f_E \in \mathbf{F}_E = \mathbf{F}_W \cap (\mathbf{F}_V \cup \mathbf{F}_I)$, where $\mathbf{F}_W$, $\mathbf{F}_V$ and $\mathbf{F}_I$ denote the sets of features extracted from $R_W$, $R_V$ and $R_I$ respectively.

Intuitively, the optimal imputation model needs to be first identified over $R_V$, then deployed to fill the missing values in $R_I$. In other words, $R_V$ and $R_I$ should share the same feature space ($\mathbf{F}_V \cup \mathbf{F}_I$). Since the evidential supports are estimated over $R_W$, we define the set of effective features, $\mathbf{F}_E$, as $\mathbf{F}_W \cap (\mathbf{F}_V \cup \mathbf{F}_I)$. It can be observed that the features other than effective ones cannot contribute to the reasoning process, thus can be filtered out. The algorithm of computing the GFD matrix is sketched in Algorithm 1 . Lines 1–4 retrieve effective features. The function `featureSet`($R$, $n$) retrieves all the features of k-grams

**Algorithm 1:** Computing the GFD matrix.

```
Input  : $R$: the tuple set;
         $n$: the maximal length of feature grams;
Output: $\mathbf{W}$: the GFD matrix.
1  $\mathbf{F}_W \leftarrow \texttt{featureSet}(R_W, n)$;
2  $\mathbf{F}_V \leftarrow \texttt{featureSet}(R_V, n)$;
3  $\mathbf{F}_I \leftarrow \texttt{featureSet}(R_I, n)$;
4  $\mathbf{F}_E \leftarrow \mathbf{F}_W \cap (\mathbf{F}_V \cup \mathbf{F}_I)$;
5  Initialize the matrix $\mathbf{C}$;
6  for each tuple $r_i$ in $R_W$ do
7      $\mathbf{F}_i \leftarrow \texttt{featureSet}(r_i, n)$;
8      for each feature $f_j \in \mathbf{F_E} \cap \mathbf{F}_i$ do
9          $\mathbf{C}[f_j, B(r_i)]$++;
10 $\mathbf{W} \leftarrow \texttt{normalize}(\mathbf{C})$;
11 return ($\mathbf{W}$)
```

($k \leq n$) from $R$. Lines 6–9 iterate over the tuples in $R_W$ and update the co-occurrence matrix **C** by counting the concurrences of $f_j$ and $b_i$, where $b_i \in Dom(B)$. Line 10 normalizes the matrix **C** row-wisely into **W**. Table 4 has shown a toy example of a GFD matrix.

On its time complexity, we have the following theorem:

**Theorem 2.** *Suppose that k-grams are extracted as features where k is a constant. The time complexity of Algorithm 1 can be represented by* $\mathbf{O}(N_F \cdot |R|)$, *in which* $N_F$ *denotes the number of extracted features, and* $|R|$ *denotes the cardinality of R.*

**Proof.** Features are extracted from the attribute values (except at $B$) over all tuples from $R$, i.e., $R_W + R_V + R_I$, thus the process of effective feature extraction takes the complexity of $\mathbf{O}(|R|)$. Updating matrix **C** takes $\mathbf{O}(N_F \cdot |R_W|)$, which is the same order of $\mathbf{O}(N_F \cdot |R|)$. Normalization for **C** takes $\mathbf{O}(N_F \cdot N_B)$, in which $N_B$ denotes the number of distinct $B$ values in $R$. In all, Algorithm 1 takes the complexity of $\mathbf{O}(N_F \cdot |R|)$. □

### 5.2. Identifying optimal model

We first present a procedure that returns the minimal match probability threshold $\theta$ and the number of imputed tuples given a combination of parameters of $(\lambda_i, \tau_j)$ and a user-specified precision level $\phi$. It is detailed in Algorithm 2 . Line 1

**Algorithm 2:** imputedSet$(\mathbf{W}, \lambda_i, \tau_j, \phi, R_V)$.

```
1  $\mathbf{L}_\mathcal{M} \leftarrow$ max(softmax$(\mathbf{W}, \lambda_i, \tau_j, R_V)$);
2  $\mathbf{L}_\mathcal{M} \leftarrow$ sort$(\mathbf{L}_\mathcal{M})$;
3  $N_\mathcal{M} \leftarrow m$;
4  $N_r \leftarrow 0$;
5  $\theta^* \leftarrow 1$;
6  for $e_k \in \mathbf{L}_\mathcal{M}$ $(1 \le k \le m)$ do
7      if $b_k^* = B(e_k)$ then
8          $N_r \leftarrow N_r + 1$;
9  $\phi^* \leftarrow \frac{N_r}{m}$;
10 if $\phi^* \ge \phi$ then
11     for $e_k \in \mathbf{L}_\mathcal{M}$ $((m+1) \le k \le |\mathbf{L}_\mathcal{M}|)$ do
12         $N_\mathcal{M} = N_\mathcal{M} + 1$;
13         if $b_k^* = B(e_k)$ then
14             $N_r \leftarrow N_r + 1$;
15         if $\frac{N_r}{N_\mathcal{M}} \ge \phi$ then
16             $\theta^* = \theta(e_k)$;
17         else
18             break;
19 else
20     $N_\mathcal{M} = 0$;
21 return $(N_\mathcal{M}, \theta^*)$
```

selects the candidate with the maximal match probability for each tuple in $R_V$. Line 2 sorts the tuples in $R_V$ by the match probabilities of their filling candidates. The variable of $N_\mathcal{M}$ at Line 3 counts the total number of imputations and $N_r$ at Line 4 counts the number of correct imputations. To prevent the iterative process from terminating too early due to a small number of incorrect imputations, lines 6–8 initialize the set of imputed tuples. At Line 7, $b_k^*$ represents the filling candidate chosen for $e_k$. Line 9 computes the precision level of the initialization set. Lines 11–18 iteratively impute the tuples in $\mathbf{L}_\mathcal{M}$. If the resulting precision level exceeds the level of $\phi$ (Line 15), it continues to impute the next tuple; otherwise, the iterative process terminates (Line 18).

The brute-force algorithm for identifying the optimal model is sketched in Algorithm 3 . It searches over the parameter

**Algorithm 3:** Brute-force strategy.

```
Input  : $\mathbf{W}$: GFD matrix;
         $\mathbf{R}_V$: the validation set of tuples ;
         $\phi$: the precision requirement;
Output: $\mathcal{M}(\lambda^*, \tau^*, \theta^*)$.
1  $\mathbf{\Lambda} \leftarrow (\lambda_1, \cdots, \lambda_n)$;
2  $\mathbf{\Omega} \leftarrow (\tau_1, \cdots, \tau_m)$;
3  $N_{opt} \leftarrow 0$;
4  for $(\lambda_i, \tau_j) \in \mathbf{\Lambda} \times \mathbf{\Omega}$ do
5      $(N_\mathcal{M}, \theta) \leftarrow$ imputedSet$(\mathbf{W}, \lambda_i, \tau_j, \phi, R_V)$;
6      if $\mathbf{N}_\mathcal{M} > N_{opt}$ then
7          $\lambda^* \leftarrow \lambda_i$;
8          $\tau^* \leftarrow \tau_j$;
9          $\theta^* \leftarrow \theta$;
10         $N_{opt} \leftarrow N_\mathcal{M}$;
11 return $\mathcal{M}(\lambda^*, \tau^*, \theta^*)$;
```

space $\boldsymbol{\lambda} \times \boldsymbol{\tau}$ in a brute-force manner. On the time complexity of Algorithm 3, we have the following theorem:

**Theorem 3.** *The time complexity of Algorithm 3 can be represented by $\mathbf{O}(N_\lambda \cdot N_\tau \cdot N_V(N_F + ln(N_V)))$, in which $N_\lambda$ and $N_\tau$ denote the cardinalities of the search space of $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ respectively, $N_F$ denotes the number of features and $N_V$ denotes the number of tuples in the validation set $R_V$.*

**Proof.** The process of computing the match probabilities over $r_i \in R_V$ takes $\mathbf{O}(N_V \cdot N_F)$. The operation of sorting the tuples in $R_V$ by match probabilities takes $\mathbf{O}(N_V \cdot \ln(N_V))$. The process of identifying the optimal threshold $\theta$ takes $\mathbf{O}(N_V)$. Therefore, Algorithm 2 takes $\mathbf{O}(N_V \cdot (N_F + \ln(N_V)))$. Since Algorithm 3 identifies the optimal model by running Algorithm 2 $N_\lambda \cdot N_\tau$ iterations, it thus takes $\mathbf{O}(N_\lambda \cdot N_\tau \cdot N_V(N_F + \ln(N_V)))$. □

### 5.3. Heuristic search approach

We also provide a heuristic approach for identifying the optimal model. It is more efficient than the brute-force approach but can achieve only a suboptimal solution. Similar to the popular Hill-Climbing algorithm, it starts with a random initialization and searches the optimal model greedily. We sketch its details in Algorithm 4 . Lines 1–6 first make the necessary initializations, and Lines 7–38 then search for the optimal model greedily by exploring neighbor regions with step size $\Delta$, until no further improvement can be made. On its time complexity, the worst case takes the same order as Algorithm 3, i.e. $\mathbf{O}(N_\lambda \cdot N_\tau \cdot N_V(N_F + \ln(N_V)))$, while the average case is much lower at $\mathbf{O}((N_\lambda + N_\tau) \cdot N_V(N_F + \ln(N_V)))$.

It is worthy to point out that in the heuristic algorithm, the reachability to the global optimum is completely determined by the initial state in the parameter space, and running the heuristic algorithm more rounds can reduce the risk of ending with local optima. We will empirically evaluate the performance of the heuristic approach in Section 6.5.

## 6. Experiments

This section reports our experimental evaluation, which is organized as follows: Section 6.1 describes the experimental setup. Section 6.2 evaluates the effectiveness of the proposed quality control mechanism. Section 6.3 compares the proposed framework with the state-of-the-art alternatives. Section 6.4 evaluates the scalability. Finally, Section 6.5 evaluates the performance of the heuristic approach for identifying the optimal model.

### 6.1. Experimental setup

Our experimental evaluations are conducted over the following two real-world relational datasets:

1. *CiteSeer*. The dataset records a collection of research papers on computer science. It contains around 50 thousand records. The missing values are located at the Journal attribute.
2. *Hotel*. The dataset contains the information of around 50 thousand hotels over the world. It consists of the attributes, Name, Address, City, State, Country, PostalCode, Airport and Currency. The missing values are located at the Airport attribute.

We select these two datasets to represent two different application scenarios. In *CiteSeer*, a very limited number of tuples have the equivalent or highly similar values on the dominant attributes. In comparison, the *Hotel* dataset has many more tuples sharing values on the dominant attributes. On both datasets, we extract both 1-grams and 2-grams as features. Following the same line of evaluating data repairing techniques by artificially injecting errors, we randomly remove values as missing data. By default, $R_I$, which denotes the set of tuples with missing values, is set to contain 10% of the tuples in a test dataset $R$. However, we also evaluate the performance of different techniques under the circumstances with various missing rates. We partition $R_C$ into $R_W$ and $R_V$, in which $R_V$ is set to contain 20% of the tuples in $R$ and the rest of tuples in $R_C$ constitute the set $R_W$.

The experiments were implemented in Python on a 64-bit Ubuntu platform. They were run on a PC with 2.3 GHz CPU and 16GB RAM. On scalability evaluation, we run each program three times and report the averaged runtime. We observe that the time difference between different runs does not exceed 5% of the runtime. The datasets and our implementations have been made publicly available[3].

### 6.2. Quality control

We first evaluate the performance of the quality control mechanism in the default setting in Section 6.2.1. By default, the missing rate is set to 10% and the confidence level is set to 0.9, or $\gamma = 0.9$. The parameter search spaces for $\lambda$ and $\tau$ are set to $1.0 \le \lambda \le 3.0$ with step size 0.5 and $1 \le \tau \le 5$ with step size 1 respectively. Then we evaluate its performance under various missing rates and various dataset sizes in Section 6.2.2 and in Section 6.2.3, respectively.

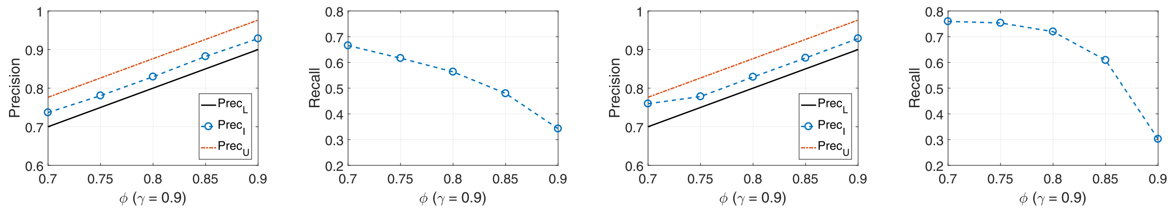
**Algorithm 4:** Heuristic search algorithm.

```
Input  : $\mathbf{W}$: GFD matrix;
         $\mathbf{R}_V$: the validation set of tuples;
         $\phi$: the precision requirement;
Output: $\mathcal{M}(\lambda^*, \tau^*, \theta^*)$.
1  $\mathbf{\Lambda} \leftarrow (\lambda_1, \cdots, \lambda_n)$;
2  $\mathbf{\Omega} \leftarrow (\tau_1, \cdots, \tau_m)$;
3  $(N_{opt}, \theta_{opt}) \leftarrow$ imputedSet($\mathbf{W}, \lambda_i, \tau_j, \phi, R_V$);
4  $\lambda_i \leftarrow randomSelect(\mathbf{\Lambda})$;
5  $\tau_j \leftarrow randomSelect(\mathbf{\Omega})$;
6  $nextIteration \leftarrow$ true;
7  while $nextIteration$ do
8      $nextIteration \leftarrow$ false;
9      $(N_{\lambda_i-\Delta,\tau_j}, \theta_{\lambda_i-\Delta,\tau_j}) \leftarrow$ imputedSet($\mathbf{W}, \lambda_i - \Delta, \tau_j, \phi, R_V$);
10     $(N_{\lambda_i+\Delta,\tau_j}, \theta_{\lambda_i+\Delta,\tau_j}) \leftarrow$ imputedSet($\mathbf{W}, \lambda_i + \Delta, \tau_j, \phi, R_V$);
11     $(N_{\lambda_i,\tau_j-\Delta}, \theta_{\lambda_i,\tau_j-\Delta}) \leftarrow$ imputedSet($\mathbf{W}, \lambda_i, \tau_j - \Delta, \phi, R_V$);
12     $(N_{\lambda_i,\tau_j+\Delta}, \theta_{\lambda_i,\tau_j+\Delta}) \leftarrow$ imputedSet($\mathbf{W}, \lambda_i, \tau_j + \Delta, \phi, R_V$);
13     if $N_{\lambda_i-\Delta,\tau_j} > N_{opt}$ then
14         $\lambda_{opt} \leftarrow \lambda_i - \Delta$;
15         $\tau_{opt} \leftarrow \tau_j$;
16         $\theta_{opt} \leftarrow \theta_{\lambda_i-\Delta,\tau_j}$;
17         $N_{opt} \leftarrow N_{\lambda_i-\Delta,\tau_j}$;
18         $nextIteration \leftarrow$ true;
19     if $N_{\lambda_i+\Delta,\tau_j} > N_{opt}$ then
20         $\lambda_{opt} \leftarrow \lambda_i + \Delta$;
21         $\tau_{opt} \leftarrow \tau_j$;
22         $\theta_{opt} \leftarrow \theta_{\lambda_i+\Delta,\tau_j}$;
23         $N_{opt} \leftarrow N_{\lambda_i+\Delta,\tau_j}$;
24         $nextIteration \leftarrow$ true;
25     if $N_{\lambda_i,\tau_j-\Delta} > N_{opt}$ then
26         $\lambda_{opt} \leftarrow \lambda_i$;
27         $\tau_j \leftarrow \tau_j - \Delta$;
28         $\theta_{opt} \leftarrow \theta_{\lambda_i,\tau_j-\Delta}$;
29         $N_{opt} \leftarrow N_{\lambda_i,\tau_j-\Delta}$;
30         $nextIteration \leftarrow$ true;
31     if $N_{\lambda_i,\tau_j+\Delta} > N_{opt}$ then
32         $\lambda_{opt} \leftarrow \lambda_i$;
33         $\tau_j \leftarrow \tau_j + \Delta$;
34         $\theta_{opt} \leftarrow \theta_{\lambda_i,\tau_j+\Delta}$;
35         $N_{opt} \leftarrow N_{\lambda_i,\tau_j+\Delta}$;
36         $nextIteration \leftarrow$ true;
37     $\lambda_i \leftarrow \lambda_{opt}$;
38     $\tau_j \leftarrow \tau_{opt}$;
39 return $\mathcal{M}(\lambda_i, \tau_j, \theta_{opt})$;
```

### *6.2.1. Precision guarantee*

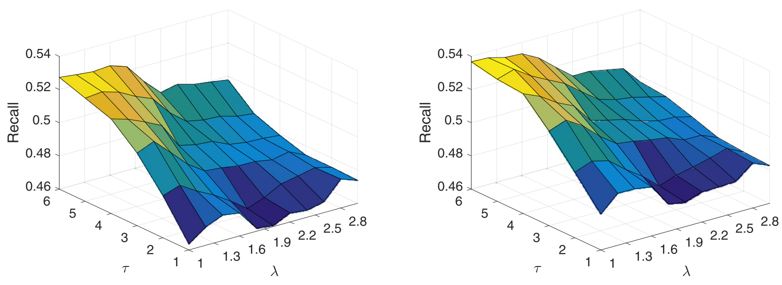
The performance of quality control on both datasets, given different precision requirements, are presented in Fig. 4(a) and (c), in which the X axis represents the user-specified precision level, the line of $Prec_I$ denotes the achieved precision level on $R_I$, and the lines of $Prec_L$ and $Prec_U$ denote its estimated lower and upper bounds respectively. It can be observed that at all the specified levels, the trained model succeeds to meet the precision requirement. As shown in Fig. 4(b) and (d),

[3] http://www.wowbigdata.com.cn/RDI/index.html.



(a) Precision (*CiteSeer*) (b) Recall (*CiteSeer*) (c) Precision (*Hotel*) (d) Recall (*Hotel*)

**Fig. 4.** Quality control on *CiteSeer* and *Hotel*.

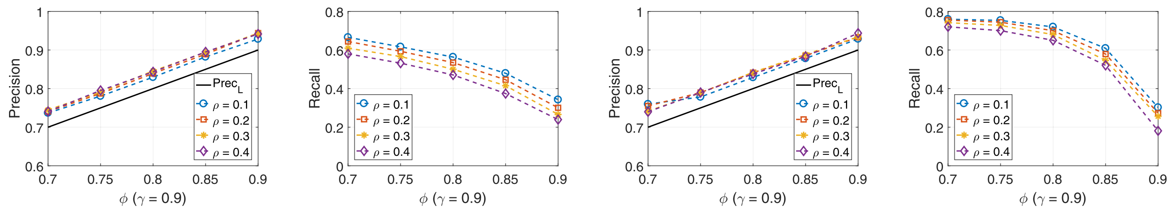


(a) Recall on $R_V$ (b) Recall on $R_I$

**Fig. 5.** Recall maximization on *CiteSeer*.

**Table 5**
Performance comparison on $R_V$ vs $R_I$ within *CiteSeer*.

| $\phi$ | Precision | | Recall | |
|---|---|---|---|---|
| | $R_V$ | $R_I$ | $R_V$ | $R_I$ |
| 0.7 | 0.7382 | 0.7368 | 0.667 | 0.6656 |
| 0.75 | 0.7883 | 0.7806 | 0.6211 | 0.6164 |
| 0.8 | 0.8383 | 0.8295 | 0.5646 | 0.5634 |
| 0.85 | 0.8884 | 0.8822 | 0.4719 | 0.4794 |
| 0.9 | 0.9383 | 0.9284 | 0.3391 | 0.3424 |



(a) Precision (*CiteSeer*) (b) Recall (*CiteSeer*) (c) Precision (*Hotel*) (d) Recall (*Hotel*)

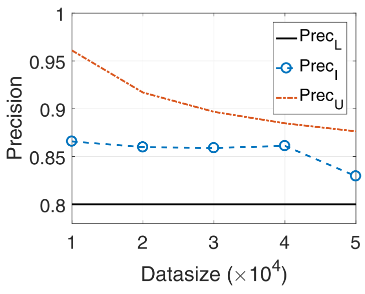
**Fig. 6.** Quality control with various missing rates.

the achieved recalls decrease with the specified precision level as expected. These experimental results clearly demonstrate the effectiveness of the quality control mechanism in enforcing precision requirement.

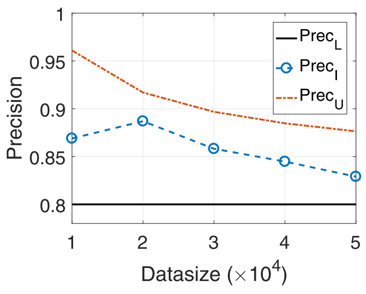
To gain better insight into the evaluation results presented in Fig. 4, we empirically investigate the effectiveness of the proposed model-training technique. We visualize the recall levels achieved by the models with various combinations of $\lambda$ and $\tau$ on $R_V$ and $R_I$ in Fig. 5. It can be observed that the performance variation patterns of the parameterized model on $R_V$ and $R_I$ are very similar: a parameter combination that achieves high recall on $R_V$ usually results in similarly high recall on $R_I$. We also compare the performance of the trained model on $R_V$ and $R_I$. The detailed results on *Citeseer* are presented in Table 5. The results on *Hotel* are similar, thus omitted here. It can be observed that the trained model achieves highly similar performance on $R_V$ and $R_I$ at all the specified precision levels.

### 6.2.2. *Varying missing rate*

On both datasets, we vary the miss rate from 0.1 to 0.4, $\rho \in \{0.1, 0.2, 0.3, 0.4\}$. The detailed results are presented in Fig. 6, in which the line of *Prec$_L$* denotes the specified precision level. It can be observed that with all the missing rates, the trained model succeeds to meet the precision requirement. As expected, the achieved recall decreases with the specified precision



(a) Precision (*CiteSeer*)



(b) Precision (*Hotel*)

**Fig. 7.** Quality control with various dataset sizes.

level. It can also be observed that the achieved recall tends to decrease with the missing rate. Higher missing rate means more tuples in $R_I$ and less tuples in $R_W$. Since less tuples are available for GFD computation, more tuples in $R_I$ would lack sufficient evidential support. However, it is worthy to point out that with the missing rate increased from 10% to 40%, the achieved recall level decreases only slightly (by less than 10%).

These experimental results demonstrate that the proposed mechanism of quality control can work reliably under various missing rates.

### *6.2.3. Varying dataset size*

To generate the test datasets with various sizes, we randomly choose tuples from the original dataset. We vary the dataset size from 10,000 to 50,000. We set the missing rate to 10% and the required precision level to 0.8, $\phi = 0.8$. The detailed evaluation results are presented in Fig. 7. It can be observed that the trained model manages to meet the precision requirement on all the test datasets. As expected, the estimated upper bound of precision ($Prec_U$) decreases with the dataset size, and the achieved precision also tends to decrease (but with some exceptions).

These experimental results demonstrate the performance robustness of the proposed mechanism with regard to dataset size. Additionally, it can clearly benefit from the rich training examples provided by a large-sized dataset.

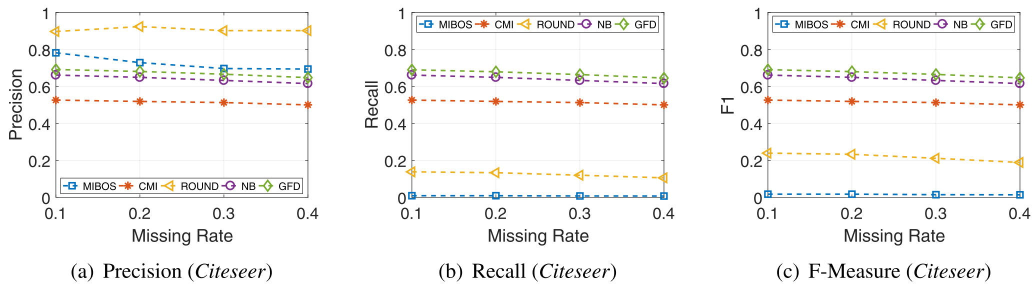
## *6.3. Comparative evaluation*

We compare the proposed framework, denoted by GFD, with the state-of-the-art alternatives, which include MIBOS [36], CMI [41], ROUND [31] and the bayesian approach [12] (denoted by NB). MIBOS is a classical technique based on value equality on the dominant attributes. CMI is a typical technique considering imputation between most similar records. ROUND enriches similarity neighbors by tolerating small variations in similarity rules such that more fillings can be acquired. NB is a popular classification-based technique. We implemented the NB approach based on the scikit-learn project [25].

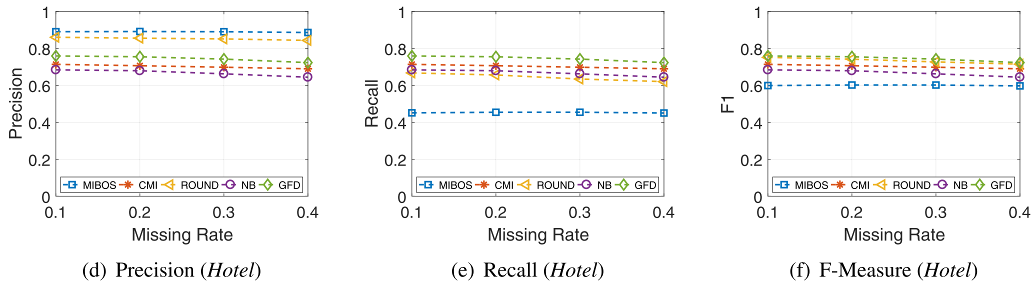
Since no precision requirement is specified, the GFD approach imputes all the tuples having missing values and at least one effective feature. The comparative results on both datasets with various missing rates are presented in Fig. 8, in which performance is measured by the metrics of precision, recall and $F_1$. On *CiteSeer*, it can be observed that MIBOS achieves high precision but has very low recall. MIBOS considers imputation only between the tuples with equivalent values on the dominant attributes. However, *Citeseer* has very few pairs of tuples sharing the same values on the dominant attributes, Author and Title. Compared with MIBOS, CMI achieves overall better performance. Instead of imputing based on value equality, CMI imputes a missing value at a tuple by considering the tuple's most similar neighbor. Therefore, it should be not surprising that CMI can fill in more missing values than MIBOS. It can also be observed that ROUND achieves considerably better performance than MIBOS. By accommodating small variances to attribute values and relaxing the neighbor definition, it does allow more missing values to be filled. ROUND also achieves higher precision than CMI. Unlike CMI, ROUND requires users to set a proper threshold on value similarity for identifying neighbors. It can therefore fill in missing values with more accuracy. On *CiteSeer*, NB achieves better performance than both CMI and ROUND. Only a few tuples in the *CiteSeer* dataset share identical or highly similar values on the dominant attributes. In such case, the classification-based approach can usually beat the NN-based approach on performance. Finally, it can be clearly observed that GFD achieves both high precision and high recall. It beats all the alternatives. Specifically, it outperforms the NN-based techniques by comfortable margins on the $F_1$ metric.

The evaluation results on *Hotel* are similar. GFD achieves overall better performance than all the alternatives. It can be observed that the NN-based techniques perform better on *Hotel* than on *CiteSeer*. In the *Hotel* dataset, there are many tuples with identical values at the dominant attributes, City and PostalCode. Therefore, the NN-based techniques can achieve higher precision and recall on *Hotel* than on *Citeseer*. It is interesting to point out that compared with the results on *Citeseer*, even though GFD outperforms the NN-based techniques by smaller margins, it outperforms NB by more considerable ones.

Observing that the NN-based techniques (e.g., MIBOS, CMI and ROUND) can benefit from the presence of clusters of duplicates, we also evaluate the performance of different techniques on the datasets with various tuple duplication rates.



(a) Precision (*Citeseer*) (b) Recall (*Citeseer*) (c) F-Measure (*Citeseer*)



(d) Precision (*Hotel*) (e) Recall (*Hotel*) (f) F-Measure (*Hotel*)

**Fig. 8.** Comparative evaluation results.

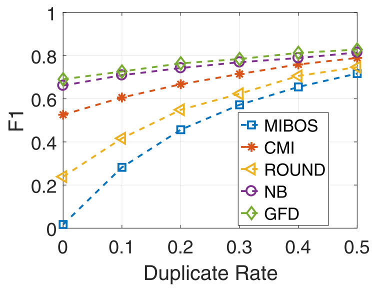


**Fig. 9.** Comparative evaluation with various tuple duplication rates.

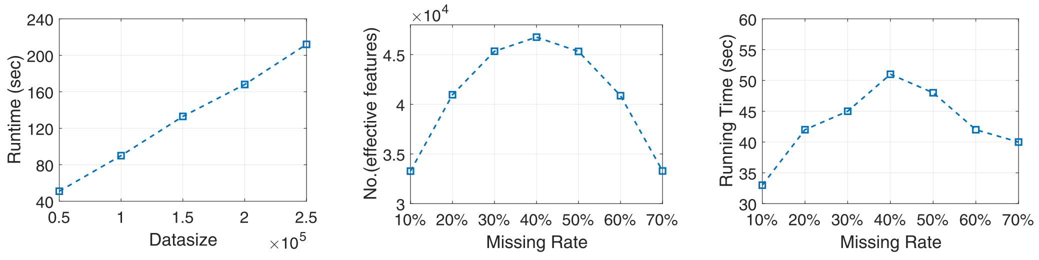
The test datasets are generated by randomly choosing a specified percentage of tuples (between 10% and 50%) in the *Citeseer* dataset and duplicating each of them. The comparative results measured on the $F_1$ metric are presented in Fig. 9. It can be observed that compared with NB and GFD, the NN-based techniques do benefit more from the presence of duplicate tuples. As the duplication rate increases from 10% to 50%, their performance improves more dramatically than that of NB and GFD. However, it can be observed that GFD consistently outperforms all the alternatives at all the duplication rates.

These experimental results show that even without a specified precision requirement, the proposed GFD approach consistently outperforms the existing alternatives on the datasets with various characteristics. They bode well for GFD's performance robustness in practical scenarios.

## 6.4. Scalability

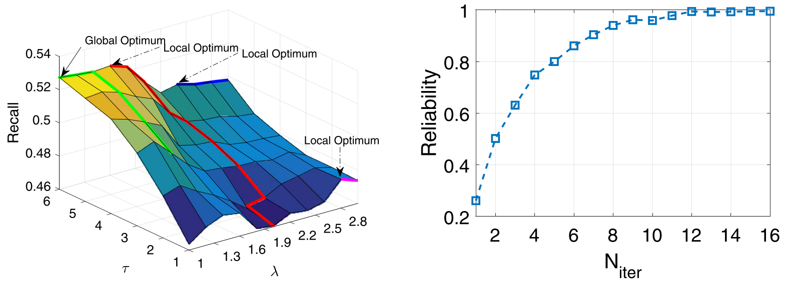
We evaluate the scalability of the GFD approach on the *Citeseer* dataset. The results on *Hotel* are similar, thus omitted here. The evaluation results are presented in Fig. 10(a), in which the runtime includes the time to compute the GFD matrix and train the imputation model. The missing rate is set to be 10%. It can be observed that the consumed runtime increases nearly linearly with the dataset size. This observation is consistent with the complexity analysis result presented in Section 5.

We also evaluate the efficiency of the GFD approach under various missing rates. We set the dataset size to 50,000 and vary the missing rate from 10% to 70%. Fig. 10(b) and (c) report the size of the effective feature set $\mathbf{F}_E$ and the runtime respectively. It can be observed that the size of the effective feature set $\mathbf{F}_E$ initially increases with missing rate, and then gradually decreases after the missing rate exceeds 40%. The runtime variation with missing rate follows a similar pattern.



(a) Runtime vs. Datasize (b) Size of $\mathbf{F}_E$ vs. Missing rate (c) Runtime vs. Missing Rate

**Fig. 10.** Scalability evaluation.



(a) Heuristic Search over $R_V$ (b) Reliability over Iterations

**Fig. 11.** Evaluation of heuristic search over *Citeseer*.

## 6.5. *Heuristic search approach*

This subsection evaluates the performance of the heuristic search approach for identifying the optimal imputation model. We run a heuristic search with 4 random initial states and plot their tracks in Fig. 11(a). It can be observed that only a few initial states (e.g., the states along the green line) can end with the global optimum while others can only end with local optima (e.g., the states along the red and blue lines). In fact, only about 27% of initial states in the parameter space shown in Fig. 11(a) can achieve the global optimum via heuristic search. Although the global optimum can not be guaranteed by heuristic search in a single iteration, the risk of ending with local optima can be reduced to an arbitrary small value $\epsilon$ by running more iterations with distinct random initial states. As shown in Fig. 11(b), the *Reliability* or the probability of converging to the global optimum increases with $N_{iter}$, which denotes the number of executed iterations. The *Reliability* has exceeded 99% (over 1000 repeats) when $N_{iter}$ increases to more than 14. More details about the relationship between *Reliability* and $N_{iter}$ are however beyond the scope of this paper.

In summary, over small search space, the brute-force strategy provides a simple and effective approach for obtaining the optimal model; while over large search space, where the brute-force strategy becomes infeasible, the global optimal model can be identified *probabilistically* by running multiple iterations of the heuristic search.

## 7. Related work

To improve data quality, it is a common practice to fill missing values based on the available information in relational datasets [10]. The existing imputation techniques can be broadly classified into statistical [14,22,28], nearest neighbor (NN) [2,11,31,40] and machine learning (ML) [3,12,13,20,26,29] approaches. However, none of them provides the capability of quality control for missing value imputation.

The statistical approaches fill the missing data such that statistical properties of interest are maximally maintained. Most of them devoted themselves to impute numerical missing data. The authors of [14] proposed to fill the missing numerical (categorical) values by the mean (mode) of the corresponding attribute values. The authors of [28] proposed a regularized Expectation Maximization approach for imputing the numerical missing values. The authors of [22] proposed to impute the missing values in gas flow data by using non-equal-length granules correlation coefficient. KPCAimpute [30] proposed to impute the missing values in microarray data by Kernel PCA.

The NN-based approaches impute a record's missing value based on its neighbors. The technique of editing rule [11] imputes a missing data with the value from the neighbor record in master data. The authors of [31] sought to maximize the imputations by tolerating small value variations in identifying neighbors, but tuning the distance threshold may be a tedious task. *k*NNI [2] first identifies the *k* nearest neighbors, and then imputes the missing values by the mode or the mean of the *k* neighbors. Again, determining the value of *k* may be a tedious task. The authors of [40] suggested imputing a missing value only if the nearest neighbor is not far from it, and giving up imputation otherwise. Most of the NN-based approaches focus on filling categorical missing data. They are prone to suffering from the problem of neighbor sparsity.

The ML approaches first train a model over complete records, and then impute the missing values either by classification or regression. The authors of [20] proposed a Bayesian approach for categorical missing data imputation. The authors of [29] employed a combination of neural network and rough set theory. MissForest [32] first fits a random forest over observed data, and then imputes the missing values iteratively until the imputations converge (no more update available for imputed data matrix). In [8], a set of decision trees were first constructed over complete records, one for each attribute having missing values; then each incomplete record is assigned to the corresponding trees leaf; finally the missing values of an incomplete record are imputed based on the records within the same leaf node. The authors of [26] proposed a hybrid classification model, which combines K-means clustering with a multilayer perceptron. UBP [13] imputes missing values with unsupervised backpropagation. The authors of [3] proposed to impute deeply learned auto encoders. As pointed out in [39], the accuracies of ML-based methods tend to be low when the proportion of missing values is too large in training data.

Our proposed framework is built on the concept of general feature dependency (GFD), which is closely related to relaxed functional dependencies (RFD) [6]. However, GFD is different from RFD in two important aspects. Firstly, RFD is defined between two attributes while GFD is defined between features and attribute values. Secondly, RFD relaxes the functional dependency on the extent of the attribute comparison, whereas GFD instead relaxes feature dependency by generalizing a single value into a multiple-value distribution on the right-hand side.

It can be observed that, even with our best, there still exists some missing data that can not be imputed with high accuracy based on the information available in a relational dataset. Our work is therefore complementary to the existing work resorting to external knowledge (e.g. Web) and human intelligence for data imputation. The authors of [38] proposed a human-machine hybrid workflow for relational missing value imputation. The authors of [39] employed Bayesian network and Crowdsourcing to improve imputation accuracy and efficiency. The authors of [7,21,34] studied how to perform relational value imputation based on the Web.

At last, there also exist some work studying how to analyze data over incomplete data. The authors of [15,23,24] evaluated top-K queries over incomplete data. The authors of [18,19,23] proposed the techniques for processing the skyline query over incomplete data. The authors of [37] studied the rule acquisition problem over incomplete multi-scale decision tables. These approaches provide alternative solutions to data analysis over incomplete data when the high-quality imputations are infeasible.

## 8. Conclusion

In this paper, we propose a novel probabilistic framework for relational data imputation based on generalized feature dependency. Unlike previous techniques, which cannot enforce quality guarantees, the proposed framework enables a flexible mechanism for quality control. We have also proposed an imputation model with the precision guarantee and presented the corresponding efficient algorithms. Finally, our extensive experiments on real datasets have demonstrated that the proposed framework performs better than the state-of-the-art alternatives and most importantly, its mechanism for quality control is effective.

Although the proposed solution can enforce precision guarantee, some applications may require the more comprehensive quality requirement specified at both precision and recall fronts. Enforcing both precision and recall is more challenging and go beyond the capabilities of the existing automatic algorithms. In future work, it is interesting to investigate how to incorporate human intelligence (e.g., crowd-sourcing) into the imputation process such that both precision and recall guarantees can be enforced.

## Acknowledgements

This work is supported by the Ministry of Science and Technology of China, National Key Research and Development Program (Project Number: 2016YFB1000700); the National Natural Science Foundation of China under Grant Nos. 61732014, 61332006, 61472321, 61502390 and 61672432.

## References

[1] C. Batini, M. Scannapieco, Data Quality: Concepts, Methodologies and Techniques, Springer Publishing Company, Incorporated, 2006.
[2] G.E.A.P.A. Batista, M.C. Monard, An analysis of four missing data treatment methods for supervised learning, Appl. Artif. Intell. 17 (5–6) (2003) 519–533.
[3] B.K. Beaulieujones, J.H. Moore, Missing data imputation in the electronic health record using deeply learned autoencoders., Pac. Symp. Biocomput. 22 (2016) 207–218.
[4] Boyd, Vandenberghe, Faybusovich, Convex optimization, IEEE Trans. Automat. Contr. 51 (11) (2006) 1859–1859.
[5] A.Z. Broder, S.C. Glassman, M.S. Manasse, G. Zweig, Syntactic clustering of the web, Comput. Networks Isdn Syst. 29 (8–13) (1997) 1157–1166.
[6] L. Caruccio, V. Deufemia, V. Polese, Relaxed functional dependencies – a survey of approaches, IEEE Trans. Knowl. Data Eng. 28 (1) (2016) 147–165.
[7] Z. Chen, Q. Chen, J. Li, Z. Li, L. Chen, A probabilistic ranking framework for web-based relational data imputation, Inf. Sci. 355 (2016) 152–168.
[8] R. Deb, W.C. Liew, Missing value imputation for the analysis of incomplete traffic accident data, Inf. Sci. 339 (2016) (2014) 274–289.
[9] K. Dehnad, Density estimation for statistics and data analysis, Technometrics 29 (4) (2012) 296–297.
[10] W. Fan, Data quality: from theory to practice, Acm Sigmod. Record. 44 (3) (2015) 7–18.
[11] W. Fan, J. Li, S. Ma, N. Tang, W. Yu, Towards certain fixes with editing rules and master data, Vldb J. 21 (2) (2012) 213–238.

[12] A.J.T. Garcia, E.R. Hruschka, Naive bayes as an imputation tool for classification problems, in: International Conference on Hybrid Intelligent Systems, 2005, pp. 497–499.
[13] M.S. Gashler, M.R. Smith, R. Morris, T. Martinez, Missing value imputation with unsupervised backpropagation, Comput. Intell. 32 (2) (2016) 196–215.
[14] J.W. Grzymala-Busse, L.K. Goodwin, W.J. Grzymala-Busse, X. Zheng, Handling missing attribute values in preterm birth data sets, in: International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, 2005, pp. 342–351.
[15] P. Haghani, S. Michel, K. Aberer, Evaluating top-k queries over incomplete data streams, 2009, pp. 877–886.
[16] E. Hazan, Introduction to online convex optimization 2(3–4) (2016) 157–325.
[17] G. Hinton, N. Srivastava, K. Swersky, Overview of mini-batch gradient descent, Neural Networks for Machine Learning, 2012.
[18] M.E. Khalefa, M.F. Mokbel, J.J. Levandoski, Skyline query processing for incomplete data, in: IEEE International Conference on Data Engineering, 2016, pp. 556–565.
[19] J. Lee, H. Im, G.W. You, Optimizing skyline queries over incomplete data, Inf. Sci. s 361–362 (2016) 14–28.
[20] X.B. Li, A bayesian approach for estimating and replacing missing categorical data, J. Data Inf. Qual. 1 (1) (2009) 3.
[21] Z. Li, M.A. Sharaf, L. Sitbon, S. Sadiq, M. Indulska, X. Zhou, A web-based approach to data imputation, World Wide Web 17 (5) (2014) 873–897.
[22] Z. Lv, J. Zhao, Y. Liu, W. Wang, Data imputation for gas flow data in steel industry based on non-equal-length granules correlation coefficient, Inf. Sci. s 367–368 (2016) 311–323.
[23] X. Miao, Y. Gao, G. Chen, T. Zhang, k-dominant skyline queries on incomplete data, Inf. Sci. Int. J. 367 (C) (2016) 990–1011.
[24] X. Miao, Y. Gao, B. Zheng, G. Chen, H. Cui, Top-k dominating queries on incomplete data, IEEE Trans. Knowl. Data Eng. 28 (1) (2015) 252–266.
[25] F. Pedregosa, G. Varoquaux, A. Gramfort, A. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[26] A. Purwar, S.K. Singh, Hybrid prediction model with missing value imputation for medical data, Expert Syst. Appl. Int. J. 42 (13) (2015) 5621–5631.
[27] S. Ruder, An overview of gradient descent optimization algorithms, CoRR abs/1609.04747 (2016).
[28] T. Schneider, Analysis of incomplete climate data: estimation of mean values and covariance matrices and imputation of missing values., J. Clim. 14 (5) (2001) 853–871.
[29] N.A. Setiawan, P.A. Venkatachalam, A.F.M. Hani, Missing attribute value prediction based on artificial neural network and rough set theory, in: International Conference on Biomedical Engineering and Informatics, 2008, pp. 306–310.
[30] Y. Shan, G. Deng, Kernel pca regression for missing data estimation in dna microarray analysis, in: IEEE International Symposium on Circuits and Systems, 2009, pp. 1477–1480.
[31] S. Song, A. Zhang, L. Chen, J. Wang, Enriching data imputation with extensive similarity neighbors, Proc. Vldb Endowment 8 (11) (2015) 1286–1297.
[32] D.J. Stekhoven, P. Bühlmann, Missforest–non-parametric missing value imputation for mixed-type data., Bioinformatics 28 (1) (2012) 112–118.
[33] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
[34] Y. Tang, H. Wang, S. Zhang, H. Zhang, R. Shi, Efficient web-based data imputation with graph model, in: Database Systems for Advanced Applications, 2017, pp. 213–226.
[35] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, 2000.
[36] S. Wu, X. Feng, Y. Han, Q. Wang, Missing categorical data imputation approach based on similarity, in: IEEE International Conference on Systems, Man, and Cybernetics, 2012, pp. 2827–2832.
[37] W.Z. Wu, Y. Qian, T.J. Li, S.M. Gu, On rule acquisition in incomplete multi-scale decision tables, Inf. Sci. Int. J. 378 (C) (2016) 282–302.
[38] C. Ye, H. Wang, Capture Missing Values Based on Crowdsourcing, 2014.
[39] C. Ye, H. Wang, J. Li, H. Gao, S. Cheng, Crowdsourcing-Enhanced Missing Values Imputation Based on Bayesian Network, Springer International Publishing, 2016.
[40] S. Zhang, Parimputation: from imputation and null-imputation to partially imputation., IEEE Intell. Inf. Bull. (2008) 32–38.
[41] S. Zhang, J. Zhang, X. Zhu, Y. Qin, C. Zhang, Missing value imputation based on data clustering, in: Transactions on Computational Science I, 2008, pp. 128–138.