On-Line Imputation for Missing Values

Fengfeng Fan, Zhanhuai Li and Yanyan Wang
School of Computer Science and Engineering
Northwestern Polytechnical University, Xi'an 710129, China

***Abstract*—Missing values are widespread in many real world applications. It is often preferred to receive in real-time the high quality complete tuples, rather than an incomplete one containing null attribute values. The requirements for high quality and real-time response make the task of missing value imputation much challenging. Most of existing approaches employ *off-line* processing, thus hardly produce in real-time the high quality imputation for the *null* values. Inspired by the incremental learning approaches, we devise a model, called OL-MVI, which seeks to produce in real-time high quality candidate on-the-fly for missing values in query results. In OL-MVI, complete tuples are analysed incrementally and summarized into a compact correlation matrix, and a scoring function is devised to guide on-line imputation in real-time. Additionally, with more data being analysed, more missing values may be imputed with high quality. Our experimental evaluations on real datasets demonstrate the effectiveness of our proposed model.**

***Keywords*—data quality; missing value; on-line; real-time**

I. Introduction

To keep track with running business, data records are often generated and fed into database continuously in streaming way. Due to incomplete entries, imperfect parsers and inaccurate extraction, missing values are widespread in real applications [1]. After a user query was submitted to such streaming database, it is preferred to receive in real-time a *high quality* complete tuple rather than an incomplete one with *null* attribute values. For example, table I presents some records of bibliography from DBLP, in which the value of $t_6[\mathsf{Journal}]$ is *null*. Given a user query q that requests information for tuple t_6, it is preferred to receive in real-time a complete tuple by imputing null values with most probably correct candidates.

The goal of missing value imputation (MVI) is to fill the *null* (missing) values based on *current* available information. Recently, a range of techniques for missing value imputation were proposed [2]–[8], which can be categorized roughly into rule-based and learning-based approaches. Rule-based approaches [2], [3] fill the missing data with the values of its neighbours who share same information, while learning-based ones learn a model from training data and fill the missing data either by classification [5]–[10] or regression. Rule-based approaches have good imputation accuracy but suffer from *neighbour sparsity* problem [2]; while learning-based ones are robust to neighbour sparsity problem but difficult to enforce quality control.

To provide complete tuples to user queries, null values in query results should be imputed on-the-fly based on *current* available information from database, rather than filled beforehand. Identifying neighbours from database takes linear time with respect to data size in rule-based approach, so does the imputation; while the model just trained tends to get *obsolete* as new records flow into the running database, and model retraining from scratch may be a time consuming task. Thus both categories hardly meet the real-time requirement for imputations.

To deliver high quality complete tuples to user queries in real-time, we propose a model called OL-MVI, short for On-Line Missing Value Imputation. Inspired from incremental learning approaches, OL-MVI can analyse new complete tuples incrementally and impute null values on-the-fly in real-time. To alleviate the *neighbour sparsity* issues, features are extracted from available information, over which imputation candidates are reasoned about. Moreover, to control the imputation precision, a scoring function is devised to guide the imputation such that only those with high scores will be accepted. With more tuples having been analysed, OL-MVI gets more knowledge about the business and more missing values can be imputed with high quality.

Our major contributions can be summarized as follows:

1) We propose a model, called OL-MVI, short for On-Line Missing Value Imputation.
2) OL-MVI can analyse new tuples incrementally, summarize the information into a correlation matrix, and impute missing values based on **current** available information in real-time, even in large data volume.
3) Experimental evaluations show the effectiveness of our approach.

For simplicity of presentation, we list the frequently used notations in Table II. The rest parts of this paper are organized as follows: Section II reviews related work. Section III defines the problem of on-line missing value imputation. Section IV describes the architecture of OL-MVI. Section V discusses the mechanism of on-line imputation. Section VI presents some experimental evaluations. and finally Section VII concludes our paper.

II. Related Work

Missing data are widespread in real applications, the problem of missing value imputation has been well studied through existing works. The aim of missing value imputation is to fill the missing values with most probably correct candidates based on the available information [1].

Existing approaches can be roughly categorized into rule-based approaches [2], [3], and learning-based approaches [5], [7]–[10], most of them employ *off-line* processing, thus hardly offer high quality imputations in real-time.

978-1-5386-1937-7/17/$31.00 ©2017 IEEE

TABLE I
DBLP BIBLIOGRAPHY

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**	*null*

TABLE II
NOTATIONS

Symbol	Notation
R^t	a snapshot of relational instance in time t;
X, Y	left-hand side and right-hand side attributes respectively;
R_I	set of tuples whose values at RHS are **not** available
R_C	set of tuples whose values at RHS are available
$\mathbf{K}^t$	correlation matrix constructed by analysing tuples in R^t
$Dom(Y)$	domain of attribute Y;
r_q	a tuple from R_I;
$r_q[X]$	value of r_q at attribute X;
F	set of features extracted from $r \in R_C$
F_q	$F_q \subset F$, set of features extracted from $r_q[X]$
f_k	$f_k \in F$, a feature in F;

There also exist some other works that study how to leverage external information sources (e.g. Web) and human intelligence for missing value imputation. [11] proposed a human-machine hybrid work-flow for relational missing value imputation. [12] employed crowd-sourcing to improve imputation accuracy. [13], [14] studied how to perform relational value imputation based on Web. But few of them offer on-line imputation in real-time.

Incremental algorithms [15], [16] are frequently applied to data streams or big data, addressing issues in data availability and resource scarcity respectively. But few works focus on applying incremental algorithms into the task of on-line imputation for missing values.

III. PROBLEM STATEMENT

A. Functional Dependency

Functional dependency is a constraint between two sets of attributes in a relation, and an example is provided as follows:

$$fd_1 : [\mathsf{X}] \rightarrow [\mathsf{Y}]$$

where X and Y are left-hand side (LHS) and right-hand side (RHS) attributes respectively. Then the task of missing value imputation is to *guess* the most probably correct candidates for missing values at RHS, based on available information. For example, editing-rule [2], which is based on functional dependency, fills the missing data with the values of its neighbours who share same information at LHS, while Naive-Bayes predicator transforms available values at LHS into features and calculates the posterior probability of each candidate.

B. Discriminative Model

There are two types of models available for classification: generative model which learns the joint probability distribution $P(X, Y)$ and discriminative model which learns the conditional probability $P(Y|X)$.

Generally speaking, the functional dependency fd_1 can be treated as a special type of discriminative model: $P(Y|X) = 1$. It implies that once the value at attribute X is fixed, then the value at attribute Y is also determined.

Analogously, for any tuple $r_q \in R$, OL-MVI also employs the discriminative model to reason about the imputation candidates. That is, imputations are based on the conditional probability $P(Y|F_q)$, where F_q represents the features extracted from attribute value of $r_q[X]$.

C. Problem Statement

As discussed above, OL-MVI seeks to reason about high quality candidates for null values at RHS attribute of query results in real-time, which can be formalized as follows:

Given a snapshot of relational instance R^t in time t, and X, Y are LHS and RHS attributes in R^t respectively. The tuples in R^t are partitioned based on values at attribute Y into two disjoint subsets: $R_C = \{r|r[Y] \neq null \wedge r \in R^t\}$ and $R_I = \{r|r[Y] = null \wedge r \in R^t\}$. Then the task of OL-MVI is to *guess* a high quality candidate $y^* \in Dom(Y)$ for null value of $r_q[Y]$ based on available features F_q and correlation matrix $\mathbf{K}^t$, where $r_q \in R_I$, F_q is the features extracted from $r_q[X]$ and $\mathbf{K}^t$ represents the knowledge learnt from R^t.

In general, the imputation candidate $y^* \in Dom(Y)$ should have the maximal conditional probability, as equation 1.

$$y^* = \mathsf{argmax}_j P(y_j|F_q, \mathbf{K}^t) \tag{1}$$

Since it is difficult to compute $P(y_j|F_q, \mathbf{K}^t)$ analytically, we resort to a linear scoring function $S(y_j|F_q, \mathbf{K}^t)$ as an approximation to $P(y_j|F_q, \mathbf{K}^t)$. Thus on-line imputation for missing values based on scoring function is formalized as follows:

$$y^* = \mathsf{argmax}_j S(y_j|F_q) \tag{2}$$

where $S(\cdot)$ is a scoring function and $\mathbf{K}^t$ is omitted for uncluttered.

IV. Architecture of OL-MVI

This section will first present the architecture of OL-MVI and then talk about some ideas behind.

A. Architecture

There are two core components in OL-MVI model as shown in figure 1:

- Analyzer will *analyse* those tuples in R_C incrementally and maintain a correlation matrix $\mathbf{K}^t$, which holds the correlations between features and imputation candidates: $\mathbf{K}^t[k, j]$ taking the frequency of $\langle f_k, y_j \rangle \in F \times Dom(Y)$.
- Predicator seeks to *guess* imputation candidate y^* for null values of $r_q[Y]$, based on features F_q and correlation matrix $\mathbf{K}^t$, where features F_q are extracted from $r_q[X]$.

Example 1: As in figure 1, a tuple r_q, having $r_q[Y] = null$, is a query result which meets the criterion of query q at X. Before returning r_q to user directly, an attempt of imputation will be made **on-the-fly** for r_q, such that a new version $r_{q'}$, with $r'_q[X] = r_q[X] \wedge r'_q[Y] = y^*$, will be returned to user.

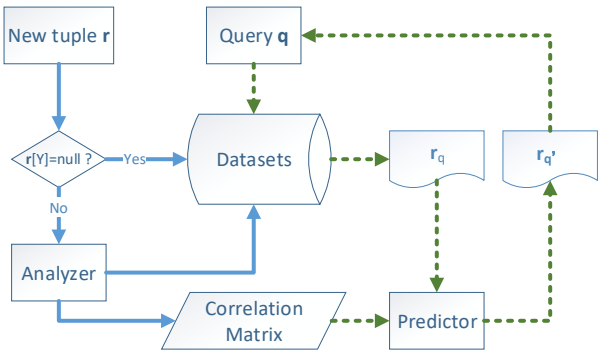

Fig. 1. Architecture of On-Line Imputation

B. Feature Extraction

In OL-MVI, the basic unit for on-line imputation is the features, over which the scores for imputation candidates are calculated. In general, each attribute value can be taken holistically as one big feature, but such processing may suffer from the *neighbour sparsity* problem [2]. To alleviate such limitation, smaller granular features (tokens or phrases) are preferred in OL-MVI. Like "database" and "query", such tokens or phrases can be extracted easily from text attribute, by open-source tools, e.g., CountVectorizer from scikit-learn project [17].

C. Scoring Function

After transforming attribute value $r_q[X]$ into features F_q, $P(Y|F_q, \mathbf{K}^t)$ can be rewritten as follows:

$$P(Y|F_q, \mathbf{K}^t) = P(Y|[f_1, f_2, \cdots, f_K], \mathbf{K}^t) \tag{3}$$

TABLE III
Evidential Supports

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

where K is the number of features included in F_q.

Since there exists no simple analytical solution to equation 3, a linear scoring function is devised as an approximation to it. The idea behind is as follows:

$$P(y_j|F_q, \mathbf{K}^t) \approx S(y_j|F_q) \tag{4}$$

Our imputation scoring function is defined on top of evidential supports from features.

Definition 1: For any feature $f_k \in F$, the evidential support for imputation candidate y_j from a single feature f_k is defined as $P(y_j|f_k)$.

Definition 2: Evidential support for y_j from features F_q can be accumulated by

$$E(y_j|F_q) = \sum_{f_k \in F_q} P(y_j|f_k) \tag{5}$$

Definition 3: Imputation score is defined as normalized evidential supports as equation 6, which can be calculated by softmax operation [18]:

$$S(y_j|F_q) = \frac{\exp(E(y_j|F_q))}{\sum_{l=1}^{L} \exp(E(y_l|F_q))} \tag{6}$$

Example 2: As shown in Table III, assume f_1 = "Query" and f_2 = "Database" are two features for consideration at the attribute of Title. We denote the attribute values at Journal, "IEEE Transactions on Knowledge & Data Engineering", "VLDB Journal" and "Information Systems" by y_1, y_2 and y_3 respectively. To fill the null value at t_6[Journal], the evidential supports can be calculated by $E(y_1|F_6) = 1.0 + 0.6 = 1.6$, $E(y_2|F_6) = 0 + 0.2 = 0.2$, and $E(y_3|F_6) = 0 + 0.2 = 0.2$. The imputation scores can be computed by Eq. 6, $S(y_1|F_6) = 0.670$, $S(y_2|F_6) = 0.165$, and $S(y_3|F_6) = 0.165$. Actually, the candidate with highest score, y_1, is just the correct imputation for t_6[Journal].

D. Quality Control

To ensure the imputation precision, a proper threshold θ should be enforced to imputation scores such that only those imputations with $S(y^*|F_q) \geq \theta$ will be accepted. Intuitively, the higher score of $S(y^*|F_q)$ is, the more probable it is imputed with the underlying true value.

$$P(y^* = y^r) \sim S(y^*|F_q) \tag{7}$$

where y^* and y^r are predicted value and the underlying true value respectively. In one words, imputation precision is positively related to the imputation score. More sophisticated techniques for quality control can be seen in [19].

V. On-Line Imputation

This section discuss the building blocks for on-line imputation, including correlation matrix, real-time imputation and incremental learning.

A. Correlation Matrix

To compute imputation scores in real-time, it is necessary to maintain some essential information about the correlations between each feature $f_k \in F$ and each candidate $y_j \in Dom(Y)$. Analyzer of OL-MVI will analyse new complete tuples in R_C, and compress the knowledge from R_C into a correlation matrix $\mathbf{K}^t$, with the element of $\mathbf{K}^t[k,j]$ taking the co-occurrence of $\langle f_k, y_j \rangle$.

Then evidential support $P(y_j|f_k)$ can be estimated easily by normalizing $\mathbf{K}^t[k,j]$ as follows:

$$P(y_j|f_k) = \frac{\mathbf{K}^t[k,j]}{\sum_{l=1}^{N_Y} \mathbf{K}^t[k,l]} \tag{8}$$

where N_Y is the cardinality of domain Y.

The correlation matrix $\mathbf{K}^t$ can be updated incrementally by analysing tuples either sequentially or in parallel, which provides OL-MVI with excellent scalability.

B. Real-Time Imputation

As in equation 6, original imputation scoring function $S(\cdot)$ takes time complexity of $O(N_q^f \cdot N_Y)$, where N_q^f and N_Y are the number of features in F_q and cardinality of $Dom(Y)$ respectively. Since both N_q^f and N_Y are independent of the number of tuples in R_C, thus the complexity of $S(\cdot)$ amounts to $O(1)$, with respect to the cardinality of R_C. The candidate with highest scores will be used for imputation, thus its complexity also amounts to $O(1)$, which means that on-line imputation can achieve to real-time response, independent of the number of tuples in R_C.

C. Incremental Learning

From figure 1, it can seen that the processes for analysing new complete tuples and imputations are separated. Benefiting from such separation, imputations triggered by the *null* values in query results r_q, will not be blocked by the analysis of new complete tuples, thus the candidates for null values can always be reasoned about **on-the-fly** based on **current** correlation matrix $\mathbf{K}^t$.

As more complete tuples having been analysed, more knowledge will be summarized into the correlation matrix $\mathbf{K}^t$, OL-MVI will also get much smarter. It implies that more missing values can imputed with high quality. For example, at early stage, only small amount of data was analysed, imputation score $S(y^*|F_q)$ based on $\mathbf{K}^t$ may fail to meet the threshold θ where $r_q \in R_I$. But as more tuples having been analysed, more knowledge available in correlation matrix $\mathbf{K}^t$, including features and their distributions, the imputation scores for true value y^r tend to grow and achieve to the threshold θ.

VI. Experimental Evaluation

In this section, we experimentally evaluate the performance of OL-MVI on real dataset *DBLP*[1]. The experimental settings are as follows: both Author and Title act as LHS attributes while Journal acts as RHS attribute. Two classical metrics were employed to evaluate the effectiveness of OL-MVI:

- Precision: $Prec = \frac{TP}{TP+FP}$, where TP and FP are the numbers of tuples *correctly imputed* and *falsely imputed* respectively.
- Recall:$Recall = \frac{TP}{N_I}$, where N_I is the total number of tuples *need to be imputed*.

All of the programs were written in Python and run on a PC with Ubuntu (16.04) OS and 16GB memory.

A. Quality Control

To verify the effectiveness of quality control, we assess the imputation precisions of OL-MVI by tuning threshold θ. In this settings, $80k$ complete tuples from R_C are analysed to construct the correlation matrix $\mathbf{K}^t$, and other $10k$ tuples from R_I are used to test the performance of OL-MVI. The imputation precisions of OL-MVI are plotted in figure 2.

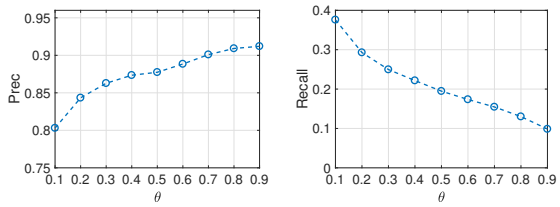

Fig. 2. Impact of θ on Precision & Recall

It can be seen that for fixed correlation matrix $\mathbf{K}^t$, imputation precision increases with threshold θ while recall decreases at the same time, which confirmed the effectiveness of mechanism for quality control.

B. Impact of Correlation Matrix

To verify the impact of correlation matrix $\mathbf{K}^t$ on imputation, we tune the number of tuples from R_C used to construct different correlation matrices. The performance of OL-MVI is evaluated on $10k$ tuples from R_I (as in subsection VI-A) with fixed threshold $\theta = 0.5$. The imputation precision and recall of OL-MVI are plotted in figure 3.

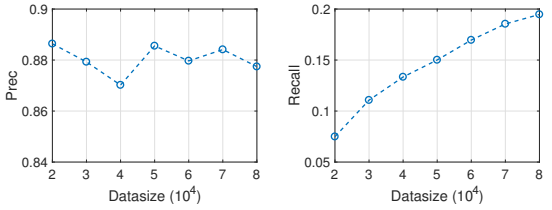

Fig. 3. Impact of $\mathbf{K}^t$ on Precision & Recall

It can be seen that, with fixed threshold $\theta = 0.5$, imputation precision is kept at a relative stable level, around 88%. But as

[1]http://dblp.uni-trier.de/xml/

more complete tuples from R_C is analysed, more knowledge is available in $\mathbf{K}^t$, thus more missing values can be imputed with high quality. For example, the recall grow from 7.5% to 19.5%, as the number of complete tuples in R_C increased from $20k$ to $80k$, which were used to construct $\mathbf{K}^t$.

C. Real-Time Imputation

To evaluate running-time performance for imputations under various correlation matrices, we will evaluate the running-time of imputations over $10k$ tuples from R_I, by tuning the number of tuples from R_C, which are used to constructed correlation matrices. The relationship between running-time for imputations and the number of tuples used for constructing correlation matrices is depicted in figure 4.

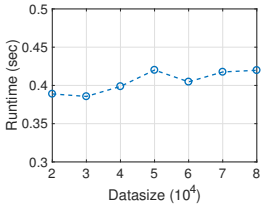

Fig. 4. Running-time vs the number of complete tuples in R_C

It can be seen that as the number of complete tuples in R_C increasing from $20k$ to $80k$, the running-time for imputation over $10k$ tuples in R_I, almost keeps stable, which is consistent with the constant time complexity, concluded from subsection V-B.

In one words, above experimental evaluations illustrate that OL-MVI is capable of producing high quality candidates in real-time, even for big data size.

VII. Conclusion

In this paper, we propose an model, called OL-MVI, for the task of on-line imputation for missing values. To achieve the on-line imputation, OL-MIV analyses the complete tuples and summarizes useful information into correlation-matrix. By employing the incremental update to the correlation matrix, it can produce high quality imputations for missing values in real-time. Additionally, the imputation quality can be controlled by tuning the threshold θ. Finally, our experimental evaluations show the effectiveness of OL-MVI.

Acknowledgment

The work was supported by the Ministry of Science and Technology of China, National Key Research and Development Program (Project Number2016YFB1000703), the National Natural Science Foundation of China under No.61332006, No.61672432 No.61472321 and No.61502390.

References

[1] W. Fan, "Data quality: From theory to practice," *Acm Sigmod Record*, vol. 44, no. 3, pp. 7–18, 2015.

[2] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, "Towards certain fixes with editing rules and master data," *Vldb Journal*, vol. 21, no. 2, pp. 213–238, 2012.

[3] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery." *Acm Transactions on Database Systems*, vol. 36, no. 3, pp. 563–574, 2011.

[4] S. Song, A. Zhang, L. Chen, and J. Wang, "Enriching data imputation with extensive similarity neighbors," *Proceedings of the Vldb Endowment*, vol. 8, no. 11, pp. 1286–1297, 2015.

[5] N. A. Setiawan, P. A. Venkatachalam, and 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.

[6] A. J. T. Garcia and E. R. Hruschka, "Naive bayes as an imputation tool for classification problems," in *International Conference on Hybrid Intelligent Systems*, 2005, pp. 497–499.

[7] X. B. Li, "A bayesian approach for estimating and replacing missing categorical data," *Journal of Data and Information Quality*, vol. 1, no. 1, p. 3, 2009.

[8] A. Purwar and S. K. Singh, "Hybrid prediction model with missing value imputation for medical data," *Expert Systems with Applications An International Journal*, vol. 42, no. 13, pp. 5621–5631, 2015.

[9] S. Zhang, "Shell-neighbor method and its application in missing data imputation," *Applied Intelligence*, vol. 35, no. 1, pp. 123–133, 2011.

[10] D. J. Stekhoven and P. Bhlmann, "Missforest–non-parametric missing value imputation for mixed-type data." *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.

[11] C. Ye and H. Wang, *Capture Missing Values Based on Crowdsourcing*, 2014.

[12] C. Ye, H. Wang, J. Li, H. Gao, and S. Cheng, *Crowdsourcing-Enhanced Missing Values Imputation Based on Bayesian Network*. Springer International Publishing, 2016.

[13] Z. Li, M. A. Sharaf, L. Sitbon, S. Sadiq, M. Indulska, and X. Zhou, "A web-based approach to data imputation," *World Wide Web*, vol. 17, no. 5, pp. 873–897, 2014.

[14] Z. Chen, Q. Chen, J. Li, Z. Li, and L. Chen, "A probabilistic ranking framework for web-based relational data imputation," *Information Sciences*, vol. 355, pp. 152–168, 2016.

[15] J. C. Riquelme, J. C. Riquelme, and J. C. Riquelme, "Incremental rule learning based on example nearness from numerical data streams," in *ACM Symposium on Applied Computing*, 2005, pp. 568–572.

[16] W. Fan, J. Li, N. Tang, and W. Yu, "Incremental detection of inconsistencies in distributed data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 6, pp. 1367–1383, 2014.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[19] Relational data imputation with quality guarantee. [Online]. Available: http://www.wowbigdata.com.cn/RDI/index.html