DOI: 10.13382/j.jemi.2017.12.022

基于正则表达式的结构化修复改进算法*

陈万志[1] 宋 剑[1] 王德建[2] 王 星[1]
(1. 辽宁工程技术大学 电子与信息工程学院 葫芦岛 125105; 2. 渤海装备辽河重工有限公司 盘锦 124010)

摘 要:针对结构化数据的清洗问题,以基于正则表达式的结构化修复(RSR)算法为基础,借鉴字符串之间编辑距离的计算思想,将违反偏序关系的边从自动机的边集中提取出来,仅对得到的边引入优先级队列来修正所对应的编辑距离,而其他边由于满足偏序关系则可直接通过递推式来计算,从而提出一种改进 RSR 算法。算法测试与分析结果表明,改进 RSR 算法在时间复杂度方面有明显优势,相对原算法的提升显著且稳定。

关键词: 数据清洗;结构化修复;正则表达式;编辑距离
中图分类号: TP312.2;TN911.72 **文献标识码**: A **国家标准学科分类代码**: 520.1010

Improved structural repairing algorithm based on regular expression

Chen Wanzhi[1] Song Jian[1] Wang Dejian[2] Wang Xing[1]
(1. School Electronics and Information Engineering, Liaoning Technical University, Huludao 125105, China;
2. China Petroleum Liaohe Equipment Company, Panjin 124010, China)

Abstract: Aiming at the structural data cleaning, an improved structural repairing algorithm based on regular expression was proposed according to calculate the edit distance between strings. Firstly, the violation partial order edge from edge set of nondeterministic finite automata was extracted, then the edit distance for edge in it was only revised by priority queue. At the same time, others edge to satisfy the partial order relation could calculate by recursive formula instead of the complex priority queue. The experimental results show that the improved algorithm not only has obvious advantage in time complexity, but also the improvement rate is significant and stable comparted with the original algorithm.

Keywords: data cleaning; structural repairing; regular expression; edit distance

0 引 言

随着移动互联网和物联网的普及,云计算和大数据分析的研究与应用不断深入,各种数据在政府和企业的管理与决策中发挥的作用越来越重要,而数据量的爆发式增长,导致海量数据未经严格预处理而大量堆积,质量参差不齐,而其质量好坏将严重影响分析决策的有效性。因此,数据质量相关问题已成为当前业界的研究热点。国内外学者对自动修复数据的方法和技术展开研究,文献[1-3]提出基于规则的修复算法,用来修改数据使之满足给定的规则集;文献[4-6]研究了从冲突的值中发现事实的算法,提出适合于对有冲突的值进行修复的基于事实的修复方法;文献[7-9]采用决策树、贝叶斯网络以及神经网络等机器学习的方法来预测丢失的数据值或做值修复等;文献[10]针对结构化数据的修复问题提出一种基于正则表达式的自动数据修复算法,首先利用正则表达式做序列修复产生正确的序列结构,然后利用关联规则和编辑距离[11]做值修复。本文在上述算法的研究和文献[12-16]的分析基础上,进一步优化改进了序列修复算法,提高了序列修复算法的效率,更加适用于海量数据的处理。

收稿日期:2017-05 Received Date: 2017-05
*基金项目:辽宁省自然基金(2015020098)、辽宁工程技术大学博士启动基金(2015-1147)资助项目

1　基于正则表达式的结构化修复(RSR)算法定义

由于正则表达式经常用来探测字符串或日期之类的序列中的错误,因此被选为结构化数据修复的首选算法基础。文献[10]提出的一种基于正则表达式的数据修复算法核心思想是使得输入序列数据以最小的修改代价满足给定的正则表达式。因此,RSR 算法的目标是将输入字符串 s 转换成满足给定的正则表达式 r 的字符串 s',且 s 与 s'之间的编辑距离最小,其形式化定义如下。

定义 1　给定正则表达式 r 和字符串 s,对每一个属于 $L(r)$的字符串 s',如果任一属于 $L(r)$的字符串 s'',都有 $ed(s,s'') \geqslant ed(s,s')$,则 s'被定义为满足正则表达式 r 的对字符串 s 的最佳修复,其中 $L(r)$表示正则表达式 r 所能描述的所有字符串,$ed(s_1,s_2)$表示字符串 s_1 和 s_2 之间的编辑距离。

由定义 1 可知所求的最小编辑距离即为输入字符串 s 与正则表达式 r 所描述的所有字符串的编辑距离的最小值,则定义该最小编辑距离为字符串 s 与正则表达式 r 的"编辑距离"。由于自动机常用来表示正则表达式匹配字符串,故可将该编辑距离定义为字符串 s 与正则表达式 r 对应的自动机 A 的"编辑距离",其形式化定义为如下。

定义 2　给定一个字符串 s 和一个自动机 A,s 和 A 的编辑距离定义为 $ed^*(s,A) = min\{ed(s,s'),s' \in L(A)\}$。其中 $L(A)$表示所有能被非确定有限自动机(non-deterministic finite automata, NFA)A 接受的字符串。

A_p 表示自动机 A 的前缀,A_p 也是一个自动机,满足如下两个条件:

1) 对任一属于 $L(A)$的字符串 s,存在 A_p 属于 $PS(A)$,使得 s_p 属于 $L(A_p)$,其中 $PS(A)$表示自动机 A 的所有前缀。

2) A_p 的状态转换图是 A 的状态转换图的子图。

当一个字符串 s 被一个自动机 A 接受,该字符串的最后一个字符所匹配的边定义为最终边,它能够描述自动机 A 与其前缀 A_p 之间的关系,$TAIL(A)$表示自动机 A 的所有最终边。

给定一个属于 $L(A)$的字符串 s,它的前缀 s_p 属于 $L(A_p)$,e 和 e_f 表示 A 的边,函数 $tr(e)$返回自动机边 e 上的转换符号,则存在属于 $TAIL(A)$的边 e_f,使得 $s_p + tr(e_f) = s$。

类比字符串之间的编辑距离,可得递归式:

$$ed^*(s_i,A) = min\begin{cases} ed^*(s_i,A_p)+1, & \text{插入} \\ ed^*(s_{i-1},A_p)+f(s[i],e_f), & \text{替换} \\ ed^*(s_{i-1},A)+1, & \text{删除} \end{cases} \quad (1)$$

其中 $f(s[i],e_f) = \begin{cases} 1, & s[i] \neq tr(e_f) \\ 0, & s[i] = tr(e_f) \end{cases}$。

2　RSR 算法描述及改进

2.1　RSR 算法描述

由于采用递归式(1)将问题划分对子问题求解的问题,故可利用动态规划的思想来解该递归式,引入矩阵 C 来存储子问题的计算结果,$C(i,e)$表示 $ed^*(s_i,A)$,其中 $e \in TAIL(A)$ 且 $s[i] = tr(e)$。则递归式(1)可等价变形为:

$$C(i,e) = min\begin{cases} C(i,e_p)+1, & \text{插入} \\ C(i-1,e_p)+f(s[i],e_f), & \text{替换} \\ C(i-1,e)+1, & \text{删除} \end{cases} \quad (2)$$

其中,$PRE(e) = \{e' \mid e'.endstate = e.startstate\}$,$e_p \in PRE(e)$。

综上所述,RSR 算法的输入为一个字符串 s 以及一个正则表达式 r 所对应的自动机 A,输出为 s 与 A 之间的最小编辑距离。通过 $dis(e)$初始化每一个 $C(0,e)$,即生成一个匹配至边 e 的字符串所需要的最小编辑次数($dis(e)$插入即可)。由于长度为 i 的字符串与空 NFA 之间的编辑距离为 i(i 次删除),故初始化 $C(i,\phi)$为 i,更详细的算法过程描述在文献[10]中均有解释,此处不再赘述。RSR 算法伪代码描述如下。

RSR 算法描述

```
Input: string s, NFA A = (P, Σ, E, q0, F).
Output: Minimal edit distance from s to A
1: C(0, e) ← dis(e), e ∈ E
2: C(i, ϕ) ← i_{i ∈ (0, len(s))}
3: for i ← 1 to len(s) do
4:   for all e ∈ E do
5:     if s[i] = tr(e) then
6:       C(i, e) ← min{C(i-1, e_p)}, e_p ∈ PRE(e)
7:     else
8:       C(i, e) ← min{C(i-1, e_p), C(i-1, e)} + 1
9:   Q ← E {Q is priority queue}
10:  while Q ≠ φ do
11:  e ← ExtractMin(Q)
12:  for all e_n ∈ NEXT(e) do
13:    if C(i, e) + 1 < C(i, e_n) then
14:      C(i, e_n) ← C(i, e) + 1
15:  C(len(s), e_t^m) ← min{C(len(s), e_t)}, e_t ∈ TAIL(A)
16:  return C(len(s), e_t^m)
```

其中 $NEXT(e) = \{e_n \mid e \in PRE(e_n)\}$

优先级队列 Q 中 $C(i,e)$越小,优先级越高

上述算法合理地避免了递推式(2)中插入操作当 $e_p = e$ 时导致的错误,可准确地计算出所需的最小编辑距离,但其存在两个问题:1)将正则表达式转换成 NFA,而不是具有更少边数的 DFA;2)引入优先队列 Q 来计算所有边的编辑距离。当处理大量的数据、更长的字符串或更复杂的正则表达式时,这种无预处理不加区分的计算必然增加算法的复杂性和运行时间,从而导致算法运行效率明显下降。

RSR 算法是一个针对序列修复问题,利用非确定自动机来计算输入串前缀与部分正则表达式的最小编辑距离的动态规划算法,算法的时间复杂度为 $O(nm^2)$,空间复杂度为 $O(nm)$(其中 m 为非确定自动机的边数,n 为输入串的长度)。对于长字符串也通过优化策略实现相应的高性能需求。同时,针对值修复问题,通过编辑距离和关联规则算法有机的融合来实现参数的自适应。文献[10]通过真实数据和合成数据的测试实验证明提出的 RSR 算法能够有效地进行数据修复。

2.2　改进的 RSR 算法

RSR 算法的核心思想主要利用计算两个字符串之间编辑距离为启发信息,将字符串与自动机之间的编辑距离划分为插入、删除或替换操作的子问题,进而利用动态规划方法解决。但是与字符串和其前缀之间的关系不同的是,自动机边的前缀并不满足严格偏序关系。如给定两条边 a、b,如果满足 $a.startstate = b.endstate$ 且 $a.endstate = b.startstate$,那么 $a \in PRE(b)$ 且 $b \in PRE(a)$,这违反了偏序的不对称属性,导致在计算递推式(2)中插入操作时不能照搬字符串之间编辑距离的计算方式。因此,与文献[10]提出的设定编辑操作优先级方式不同,本文提出的改进算法依然借鉴字符串之间编辑距离的计算思想,但将违反偏序关系的边(*bad_edges*)从自动机的边集中提取出来,仅对 *bad_edges* 中的边引入优先级队列来修正所对应的编辑距离,其他边由于满足偏序关系可直接通过递推式来计算而无需使用更复杂的优先级队列。

改进后的算法与原算法框架类似,时间复杂度并没有数量级上的改变,但改进后的算法严格地区分不同性质的边,只对 *bad_edges* 进行第 13 ~ 17 行的修正,有效地降低了时间复杂度的常数因子。*bad_edges* 在最外层的循环外进行计算,循环内第 10 ~ 17 行的计算相对于不加区分完全使用优先级队列来修正编辑距离也是个更优的处理方式,故改进后的算法开销显然优于原算法,后续的实验测试结果也证实了改进后的算法有效性和优越性。其算法的伪代码如下。

改进 RSR 算法描述

```
Input: string s, DFA A = (P, ∑, E, q0, F).
Output: Minimal edit distance from s to A
1:  C(0, e) ← dis(e), e ∈ E
2: C(i, ϕ) ← i_{i ∈ (0, len(s))}
3: bad_edges = {e | PRE(e) ∩ {e, NEXT(e)} ≠ ϕ}
4:  for i ← 1 to len(s) do
5:    for all e ∈ E do
6:      if s[i] = tr(e) then
7:        C(i, e) ← min{C(i − 1, e_p)}, e_p ∈ PRE(e)
8:      else
9:        C(i, e) ← min{C(i − 1, e_p), C(i − 1, e)} + 1
10:  if  e ∉ bad_edges then
11:  C(i,e) ← min{C(i,  e_p) + 1, C(i,e)}
12:    Q ← bad_edges {Q is priority queue}
13:   while Q ≠ ϕ do
14:  e ← ExtractMin(Q)
15:  for  all e_n ∈ NEXT(e) do
16:     if C(i, e) + 1 < C(i, e_n) then
17:       C(i, e_n) ← C(i, e) + 1
18:  C(len(s), e_t^m) ← min{C(len(s), e_t)}, e_t ∈ TAIL(A)
19:  return C(len(s), e_t^m)
```

3　算法测试与结果分析

算法测试环境为 Intel Core2 Duo CPU P7370 @ 2.00 GHz,4 GB DDR3,32 位 Windows 7 专业版,编程语言为 Python 2.7.10。算法测试流程如图 1 所示,具体完成了文献[10]的 RSR 算法和本文改进后的 MyRSR 算法的实现与对比分析。

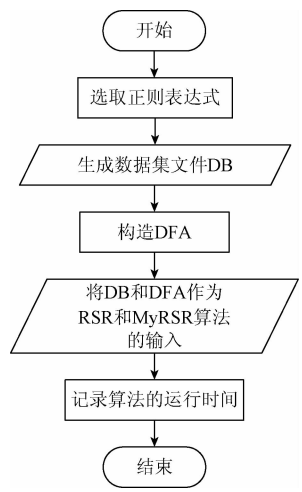

图 1　算法测试流程

Fig. 1　Flow chart of algorithm test

3.1　实验数据

为了验证大量闭包和选择等极端条件,算法测试的实验数据采用文献[10]中具有代表性的正则表达式所生成的模拟数据集,即由 Regex_1:

$a*b((cd)\mid(de)\mid(ec))*f((gh)\mid(ij)\mid(klm))*n((o(pqz)*)\mid((r\mid s)t*))((uv)\mid(wxy))*$

生成 50 个数据集文件,每个数据集文件包含 100 ~ 5 000条字符串,其中闭包表示的重复次数根据实际经验是随机分布在区间[0,10],因此生成的字符串长度均应在 50 个字符左右。

3.2　实验结果分析

以相同的数据集为实验数据进行测试时所记录 RSR 和 MyRSR 算法的运行时间为基础,分别采用算法的运行时间来对比两者的绝对优劣,采用提升比(improve rate)来对比两者的相对优劣,其中提升比定义为改进后的算法相较于原始算法提升的比例,即 $Improve\ Rate = \frac{t_{RSR} - t_{MyRSR}}{t_{RSR}}$。

从图 2 可知,两种算法的运行时间均随着数据量的增多而线性增长,但改进后的算法运行时间始终比原始算法所耗费时间少,且数据量越大这种差异越显著,这也验证了将违反偏序关系的边从自动机的边集中分离出来处理的正确性。

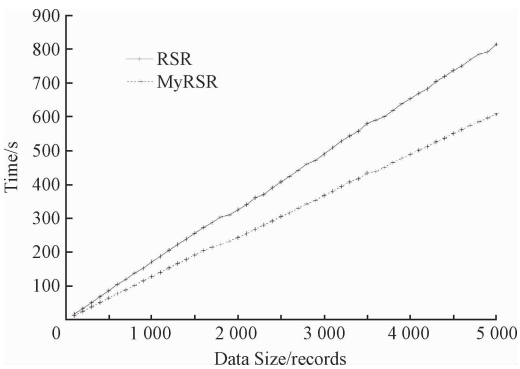

图 2　两种算法运行时间对比(Regex_1)

Fig. 2　Comparison of running times of two algorithms (Regex_1)

从图 3 可知,改进后的算法的优势(提升比)明显且稳定,始终保持在 0.25 左右,这也验证了改进后的算法的有效性。

此外,算法测试过程中还对文献[10]所给出的其他 3 条代表性正则表达式进行了对比实验,可得到相似的实验数据和结论。

由 Regex_2:

$(ab)*cde*(fgh)*((ij)*kl)*m(n*op)*((qrs)*(tu)*)*(vwxy)*z$

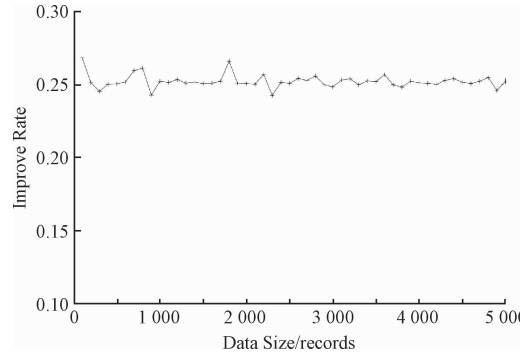

图 3　算法提升比(Regex_1)

Fig. 3　Improvement rate of algorithm (Regex_1)

生成的数据集文件为实验数据进行测试时所记录 RSR 和 MyRSR 算法的运行时间和提升比如图 4、5 所示。

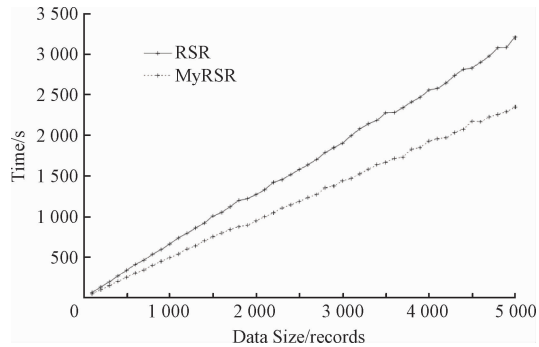

图 4　两种算法运行时间对比(Regex_2)

Fig. 4　Comparison of running times of two algorithms (Regex_2)

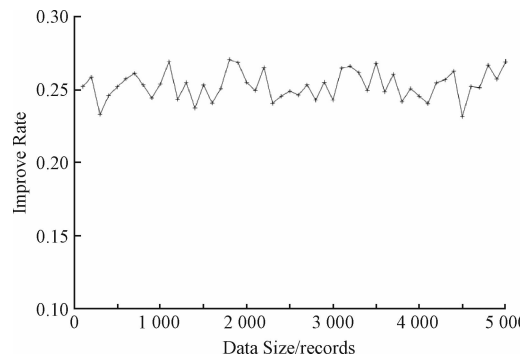

图 5　算法提升比(Regex_2)

Fig. 5　Improvement rate of algorithm (Regex_2)

Regex_3:

$((ab)\mid c)*((def)\mid(gh)*(ijk)*((lmn)*\mid(opg)*)*((rst)\mid(uvw)\mid(xyz))*$

生成的数据集文件为实验数据进行测试时所记录 RSR 和 MyRSR 算法的运行时间和提升比如图 6、7 所示。

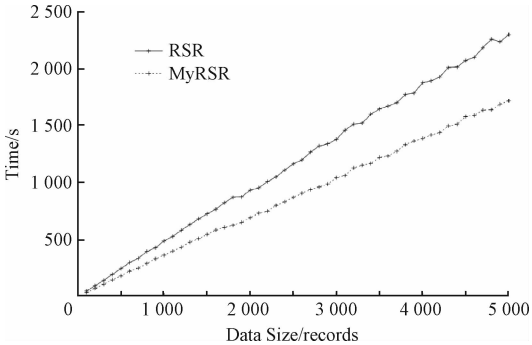

图6　两种算法运行时间对比(Regex_3)

Fig. 6　Comparison of running times of two algorithms (Regex_3)

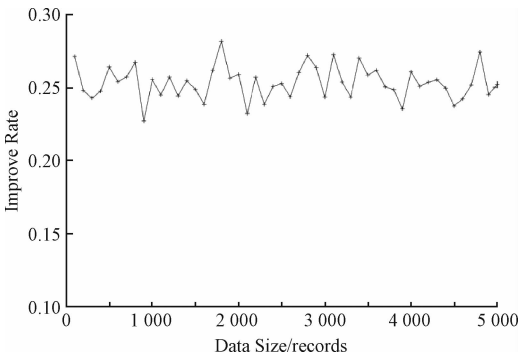

图7　算法提升比(Regex_3)

Fig. 7　Improvement rate of algorithm (Regex_3)

Regex_4:

$((ab)\mid(cd))*ef((ghi)*\mid(jk)*)((lm)*\mid(nop)*)*((qr)*s\mid t(uv)*)*w(x\mid y)*z$

生成的数据集文件为实验数据进行测试时所记录RSR和MyRSR算法的运行时间和提升比如图8、9所示。

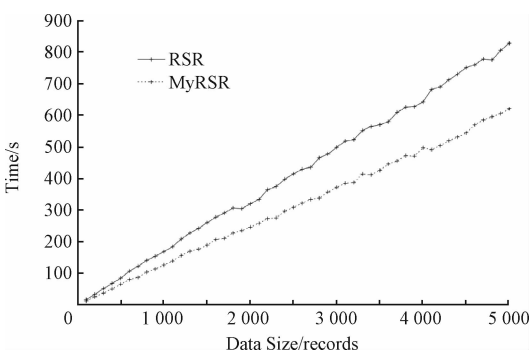

图8　两种算法运行时间对比(Regex_4)

Fig. 8　Comparison of running times of two algorithms (Regex_4)

综上所述,改进后的算法在运行时间和提升比两方面相对原始算法均有明显的优势。在相同结构化数据集

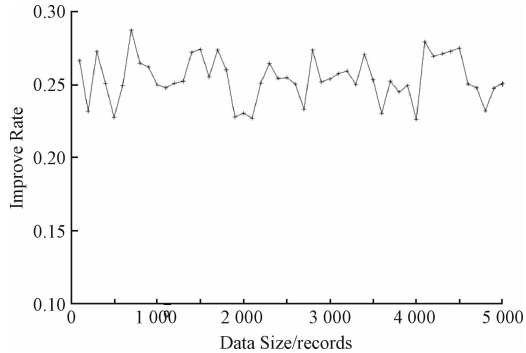

图9　算法提升比(Regex_4)

Fig. 9　Improvement rate of algorithm (Regex_4)

背景下,随着数据量的大幅增加,将自动机的边集中违反偏序关系的边分离的是有意义的,算法时间复杂度数据量级相同情况下,改进RSR算法较原算法的运行时间更短(线性斜率对比),提升比可稳定达到0.25左右(折线平均值)。

4　结　论

本文研究了结构化数据的结构修复问题,在分析RSR算法基础上提出将违反偏序关系的边从自动机的边集中分离出来处理的改进思路。算法测试实验结果表明,改进后的算法较原始算法具有运行时间短,提升比明显且稳定的优势。下一步的研究将进一步在具有实际应用背景的海量数据场景中对改进算法进行完善性测试。

参考文献

[1] ARENAS M, BERTOSSI L E, CHOMICKI J, et al. Scalar aggregation in inconsistent databases [J]. Theoretical Computer Science, 2003, 296(3): 405-434.

[2] FAN W, GEERTS F, TAN N, et al. Inferring data currency and consistency for conflict resolution[C]. IEEE International Conference on Data Engineering, 2013: 470-481.

[3] GEERTS F, MECCA G, PAPOTTI P, et al. The LLUNATIC data-cleaning framework[J]. Proceedings of the VLDB Endowment, 2013, 6(9): 625-636.

[4] DONG X L, BERTI-EQUILLE L, SRIVASTAVA D. Integrating conflicting data: The role of source dependence[J]. Proceedings of the VLDB Endowment, 2009, 2(1): 550-561.

[5] DONG X L, BERTI-EQUILLE L, SRIVASTAVA D. Truth discovery and copying detection in a dynamic world[J]. Proceedings of the VLDB Endowment, 2009, 2(1): 562-573.

[6] GALLAND A, ABITEBOUL S, MARIAN A, et al.

Corroborating information from disagreeing views[C]. ACM International Conference on Web Search and Data Mining, 2010: 131-140.

[7] LAKSHMINARAYAN K, HARP S A, GOLDMAN R P, et al. Imputation of missing data using machine learning techniques[C]. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996: 140-145.

[8] MAYFIELD C, NEVILLE J, PRABHAKAR S. ERACER: A database approach for statistical inference and data cleaning[C]. Proceedings of the ACM SIGMOD International Conference on Management of Data, 2010: 75-86.

[9] SETIAWAN N A, VENKATACHALAM P A, HAN A F M. Missing attribute value prediction based on artificial neural network and rough set theory[C]. Proceedings of the 2008 International Conference on Biomedical Engineering and Informatics, 2008: 306-310.

[10] LI Z, WANG H, SHAO W, et al. Repairing data through regular expressions[J]. Proceedings of the VLDB Endowment, 2016, 9(5): 432-443.

[11] WAGNER R A, FISCHER M J. The string-to-string correction problem[J]. Journal of the ACM, 1974, 21(1): 168-173.

[12] 曹建军,刁兴春,汪挺,等. 领域无关数据清洗研究综述[J]. 计算机科学, 2010, 37(5): 26-29.
CAO J J, DIAO X CH, WANG T, et al. Research on domain-independent data cleaning: A survey[J]. Computer Science, 2010, 37(5): 26-29.

[13] 王日芬,章成志,张蓓蓓,等. 数据清洗研究综述[J]. 现代图书情报技术, 2007, 2(12): 50-56.
WANG R F, ZHANG CH ZH, ZHANG B B, et al. A survey of data cleaning[J]. New Technology of Library and Information Service, 2007, 2(12): 50-56.

[14] 刘喜文,郑昌兴,王文龙,等. 构建数据仓库过程中的数据清洗研究[J]. 图书与情报, 2013, 153(5): 22-28.
LIU X W, ZHENG CH X, WANG W L, et al. Research on data cleaning in the process of building data warehouse[J]. Library and Information, 2013, 153(5): 22-28.

[15] 宋金玉,陈爽,郭大鹏,等. 数据质量及数据清洗方法[J]. 指挥信息系统与技术, 2013, 4(5): 63-70.
SONG J Y, CHEN SH, GUO D P, et al. Data quality and data cleaning methods[J]. Command Information System and Technology, 2013, 4(5): 63-70.

[16] 唐煜程,张明君,王浩宇,等. 基于 GPU 的三维人脸数据动态线性快速修复[J]. 电子测量与仪器学报, 2016, 30(6): 959-967.
TANG Y CH, ZHANG M J, WANG H Y, et al. Fast linear recovering algorithm for low quality 3D face data based on GPU[J]. Journal of Electronic Measurement and Instrumentation, 2016, 30(6): 959-967.

作者简介

陈万志,1977 年出生,现任辽宁工程技术大学副教授,主要研究方向为人工智能、计算机过程控制、物联网工程等。
E-mail:chenwanzhi@lntu.edu.cn

Chen Wanzhi was born in 1977. And he is an associate professor in Liaoning Technology University now. His present research interests include artificial intelligence, computer process control, and internet of things and so on.

宋剑,1990 年生,辽宁工程技术大学硕士研究生,主要研究方向为人工智能、物联网应用。
E-mail:1198513329@qq.com

Song Jian was born in 1990. And he is a M. Sc. candidate in Liaoning Technology University now. His present research interest includes artificial intelligence, and internet of things.