基于输入样本和主数据的编辑规则挖掘算法①

杨 辉, 于守健, 陈少总

(东华大学 计算机科学与技术学院, 上海 201620)

摘 要: 基于编辑规则和主数据的数据修复技术能自动地、确切地修复不一致数据, 但目前编辑规则的获取主要依靠专业人员的定义. 为了实现数据清洗全自动化, 数据规则的挖掘技术近年来成为研究热点, 针对条件函数依赖提出的挖掘算法主要有 CFDMiner, CTANE, FastCFD. 在此基础上, 扩展条件函数依赖(CFD)的定义, 在编辑规则的定义下提出了一种基于输入样本和主数据的编辑规则挖掘算法, 主要思路是从输入样本中挖掘出 CFD, 然后根据输入样本与主数据在属性上的定义域相似性求出输入样本在主数据中的对应属性, 从而形成带模式组的编辑规则, 此算法能有效地挖掘编辑规则. 且所挖掘的编辑规则按照编辑规则语义能有效地进行数据修复.

关键词: 编辑规则; 条件函数依赖; 数据清洗; 等价类划分

Method for Discovering Editing Rules From Sample Inputs and Master Data

YANG Hui, YU Shou-Jian, CHEN Shao-Zong

(School of Computer Science and Technology, Donghua University, Shanghai 201602, China)

Abstract: Data repairing based on editing rules and master data can automatically and exactly fix inconsistent data, but editing rules mainly relies on the definition by professional staff at present. To achieve data cleaning automatically in the whole process, the techniques for discovering data rules become a hot research topic in recent years. The algorithms for mining CFDs mainly involve CFDMiner, CTANE, FastCFD. Based on the above techniques, we provide a mining algorithm for editing rule, which is based on sample inputs and master data under the extension definition of CFD and the definition of edit rules. The main ideas is as below: Mining CFD from sample inputs firstly; then according to the domain similarity between input samples and master data, we can get the corresponding properties of input samples from the master data, forming editing rules with pattern group. The algorithm can effectively discover edit rules. And the mined edit rules can effectively repair the data in accordance with the semantic of the rules.

Key words: editing rules; conditional functional dependency; data cleaning; equivalence classes partitions

1 引言

基于编辑规则和主数据的数据修复[1]比基于 CFD 的数据修复[2]更有效, 主要体现在数据修复的精确性上, 且编辑规则不会引入新的错误. 基于编辑规则的清洗方案更有望被数据清洗工具所采用, 这样, 从输入样本自动挖掘编辑规则的技术是很有必要的. 事实上, 仅仅依靠领域昂贵的专业人员和漫长的手工来定义规则往往是不现实的. 正如 Gartner 所报道, 清洗规则的挖掘在商业数据质量工具中至关重要.

在实际中, 关于编辑规则挖掘关心的问题是: 给定关系模式 R 中的样本实例 r, 找到满足实例 r 的所有编辑规则集 Σ 的正则覆盖. 即逻辑上等价于 Σ 的规则集 Σ_c. 挖掘编辑规则的关键任务是挖掘 CFDs, 最后去匹配主数据中的属性, 匹配方法是将输入样本在每个属性上的定义域与主数据每个属性上的定义域进行比较, 定义域相似度满足定义的阈值即为对应的属性, 并将对应的属性保存在哈希表中, 这样我们得到 CFDs 后, 从哈希表中找到对应的属性就可以得到我们所需的编辑规则了. 为了减少冗余, 应该使所挖掘的每个 CFD 是最小的, 即非平凡的和左约简的. 挖掘

① 收稿时间:2016-07-17;收到修改稿时间:2016-09-13 [doi:10.15888/j.cnki.csa.005728]

的难点是时间复杂度高, 从实例中挖掘出 CFDs 的正则覆盖集时间复杂度就已经达到指数级了. 再者, 编辑规则需要绑定主数据中的属性, 会遇到 CFDs 挖掘中不曾遇到的困难, 根据表 1 中的实例数据 r_0 [3], 观察可以得到如下几个 CFDs, 分别表示为:

$\varphi_0:([\mathrm{CC,ZIP}]\to \mathrm{STR},(44,\_\,\|\,\_))$

$\varphi_1:([\mathrm{CC,AC}]\to \mathrm{CT},(01,908\|\mathrm{MH}))$

$\varphi_2:([\mathrm{CC,AC}]\to \mathrm{CT},(44,131\|\mathrm{EDI}))$

$\varphi_3:([\mathrm{CC,AC}]\to \mathrm{CT},(01,212\|\mathrm{NYC}))$

从上面的 CFDs 规则可以看出, 国家编码(CC)和地区编码(AC)可以决定城市, 规则 φ_0 表明在英国, 邮政编码(ZIP)可以唯一决定街道(STR). 从上面观察所得的 CFDs 可以看出有两类 CFD, 但规则左边不是最约简的, 因此通过算法能同时挖掘出这两类最小 CFDs 是具有挑战的; 另一个挑战是对 CFDs 扩展, 即确定编辑规则中来自主数据的属性集, 表 2 中的主数据, 与输入实例数据来源不同, 字段名不能一一对应起来, 例如电话号码可以表示成 PN 或 TEL, 地址可以表示成 STR 或 POST. 为了解决以上难点, 提出了能挖掘出最小的两类 CFDs 和匹配主数据形成编辑规则的算法.

表 1 Customer Relation schema R

	CC	AC	PN	NM	STR	CT	ZIP
t_1	01	908	1111111	Mike	Tree Ave.	MH	07974
t_2	01	908	1111111	Rick	Tree Ave.	MH	07974
t_3	01	212	2222222	Joe	5th Ave	NYC	01202
t_4	01	908	2222222	Jim	Elm Str.	MH	07974
t_5	44	131	3333333	Ben	High St.	EDI	EH4 1DT
t_6	44	131	4444444	Ian	High St.	EDI	EH4 1DT
t_7	44	908	4444444	Ian	Port PI	MH	W1B 1JH
t_8	01	131	2222222	Sean	3rd Str.	UN	01202

表 2 Relation schema R_m

	CC	AC	TEL	NM	POST	CT	ZIP	gender
s_1	01	908	1111111	Mike	Tree Ave.	MH	07974	Male
s_2	01	212	2222222	Joe	5th Ave	NYC	01202	Male
s_3	44	131	3333333	Ben	High St.	EDI	EH4 1DT	Male

2 相关概念

2.1 CFDs 相关概念

2.1.1 CFDs 定义

在关系 R 上的条件函数依赖 φ 的形式为 $(R:X\to A,t_p)$, 其中 $X\to A$ 是标准的 FD 形式, $X\in attr(R)$, t_p 是属性 X 和 A 上的模式组[4]. 规则的左边属性集记为 LHS(φ), 右边记为 RHS(φ). 例如 φ_0 中, ([CC,ZIP]$\to$STR) 为 FD 形式, (44,_ ‖ _)为模式组, LHS(φ_0) 为[CC,ZIP], RHS(φ_0) 为 STR.

2.1.2 CFDs 分类

如果 $t_p[A]$ 等于常数, 且任意 $B\in X$ 上的属性值 $t_p[B]$ 也都等于常数, 则该 φ 为常 CFD. 如果 $t_p[A]=\_$ 且 X 中存在属性 B 有 $t_p[B]=\_$, 则该 φ 为变 CFD[3]. 例如前面提到的 φ_0 为变 CFD, $\varphi_1,\varphi_2,\varphi_3$ 为常 CFDs.

2.1.3 CFDs 语义[5]

为了说明 CFD 的语义, 我们先在属性值上定义一个关系符 $\le$, 如果 $\eta_1=\eta_2$ 或者 η_1 为常数, η_2 为"_", 则有 $\eta_1\le\eta_2$. 例如表 1 中元组, (44,"EH4 1DT","EDI") $\le$ (44,_,_). 如果 $t_1\le t_2$ 我们就说 t_1 匹配 t_2; 如果 $t_1\le t_2$, 不 $t_2\le t_1$ 记为 $t_1\ll t_2$. 例如 (44,"EH4 1DT","EDI") $\ll$ (44,_,_). 给定规则 φ, 如果对于实例 r 中的每对元组 t_1,t_2 当 $t_1[X]=t_2[X]\le t_p[X]$ 时, 都有 $t_1[A]=t_2[A]\le t_p[A]$ 成立, 则 r|=φ. 如果定义满足 φ 的一个 r 子集为 $r_\varphi=\{t\,|\,t\in r,t[X]\le t_p[X]\}$, 那么只需考虑 r_φ 中的任意两对 t_1,t_2 当 $t_1[X]=t_2[X]$ 时, 是否有 $t_1[A]=t_2[A]$ 和 $t_1[A]\le t_p[A]$ 成立, 而不用考虑整个 r. 如果 r 满足规则集 Σ 中的每个 φ, 则 r |=Σ. 对于 $(X\to Y,t_p)$ 中 Y 为多属性集合的规则, 等价于多个规则右边只有一个属性的规则. 这里我们只考虑 RHS(φ) 为单一属性集的规则.

2.1.4 最小 CFDs

给定了模式 R 中的样本实例 r，挖掘算法的目的是找到所有 r 满足的 CFDs. 为了得到的规则集是非冗余的，仅包含最小 CFDs，首先给出最小 CFDs 的形式化定义.

对于 R 上的 CFD $\varphi=(X\to A,t_p)$，如果 $A\in X$，则为平凡的 CFD. 因为这样的规则，当 $t_p[A_L]=t_p[A_R]$ 时，所有的实例都能满足，当 $t_p[A_L],t_p[A_R]$ 为不同的常数时，所有的实例都不能满足. 故我们只考虑非平凡的 CFDs. 对于常 CFDs，如果 $Y\subset X, r\models(Y\to A,(t_p[Y]\|a))$ 则为左约简的. 对于变 CFDs，如果 $Y\subset X, r\models(Y\to A,(t_p[Y]\|\_))$ 且对任意 $t_p\ll t'_p$ 的 t'_p 有 $r\models(Y\to A,(t'_p[Y]\|\_))$ 则为左约简的. 非平凡的，左约简的 CFD 称为最小 CFD. 例如，第一部分中 φ_0 是最小 CFD；φ_1 不是最小 CFD，因为从 LHS(φ_1) 可以删掉属性 CC 后，仍有 $r_0\models(\mathrm{AC}\to\mathrm{CT},(908\|\mathrm{MH}))$.

2.1.5 频繁 CFDs

现实中的数据往往包含一些脏数据，为了排除包含错误的 CFD，只考虑那些模式组的支持度不小于阈值的 CFDs. $\varphi=(X\to A,t_p)$ 在 r 中的支持度记为 $\sup(\varphi,r)$，表示 r 中满足规则 φ 的元组数. 对自然数 $k\geq 1$，如果 $\sup(\varphi,r)\geq k$，则 CFD 为 k-频繁的. 例如 φ_1 是 3-频繁的，φ_2 是 2-频繁的.

2.2 编辑规则相关概念

2.2.1 编辑规则定义

定义在 (R,R_m) 上的编辑规则是一对 $((X,X_m)\to(A,A_m),t_p[X])$，其中:

(1) X 和 X_m 分别来自模式 R 和 R_m，$|X|=|X_m|$;

(2) $A\in R\setminus X$，$A_m\in R_m\setminus X_m$;

(3) t_p 是属性 X 上的模式组，对每一个 $B\in X$ 的属性，其值为定义域 $dom(B)$ 中的某个常数 a，或者任意值，用"_" 表示.

根据表 1 得出的四个 CFDs，我们只取 LHS(φ) 对应的属性值，结合主数据表 2，可以得到如下四个编辑规则:

$\psi_0:(([\mathrm{CC},\mathrm{ZIP}],[\mathrm{CC},\mathrm{ZIP}])\to(\mathrm{STR},\mathrm{POST}),(44,\_))$

$\psi_1:(([\mathrm{CC},\mathrm{AC}],[\mathrm{CC},\mathrm{AC}])\to(\mathrm{CT},\mathrm{CT}),(01,908))$

$\psi_2:(([\mathrm{CC},\mathrm{AC}],[\mathrm{CC},\mathrm{AC}])\to(\mathrm{CT},\mathrm{CT}),(44,131))$

$\psi_3:(([\mathrm{CC},\mathrm{AC}],[\mathrm{CC},\mathrm{AC}])\to(\mathrm{CT},\mathrm{CT}),(01,212))$

从上面的四个编辑规则可以看到来自不同数据源的模式表，代表同一实体的属性命名是存在差异的，不一定与样本模式表 R 中的属性对应. 再者，输入样本可能存在少量脏数据，为此，我们只能通过它们的定义域相似度来确定映射关系.

2.2.2 编辑规则语义

编辑规则除了具有静态语义还具有类似于匹配依赖[6](MDs)的动态语义，根据上面得到的四个编辑规则 $\psi_0-\psi_3$，如果 $t[X]\leq t_p[X]$ 且 $t[X]=t_m[X_m]$ 则 $t[A]:=t_m[A_m]$，意思是说如果待修复元组能匹配模式组且能在主数据中找到 LHS(ψ) 上相等的元组，则用该元组 A_m 属性上的值修改待修复元组 A 属性上的值.

2.2.3 等价类划分[7]

定义在属性集 X 上的等价类为一类在属性 X 上值相等的元组集，表示成 $[t]_X=\{u\in r\mid t[A]=u[A],\forall A\in X\}$. 属性 X 上所有的等价类组成的集合为属性 X 上的划分，记为 $\pi_X=\{[t]_X,t\in r\}$. 所有等价类的并集等于实例 r. 例如对于表 1 中的数据，$[t_1]_{\{\mathrm{AC}\}}=[t_2]_{\{\mathrm{AC}\}}=[t_4]_{\{\mathrm{AC}\}}=[t_7]_{\{\mathrm{AC}\}}=\{t_1,t_2,t_4,t_7\}$，$\pi_{\{\mathrm{AC}\}}=\{\{t_1,t_2,t_4,t_7\},\{t_3\},\{t_5,t_6,t_8\}\}$.

3 编辑规则挖掘算法

关于函数依赖、条件函数依赖、关联规则挖掘算法的研究，文献[8]和文献[9]将其统一为数据质量规则，并提出了数据质量规则挖掘算法 QRMiner. 考虑到输入的样本数据本来就可能存在脏数据，文献[10]还对所挖掘的数据质量规则从 3 个角度进行了评估. 本文主要是对挖掘条件函数依赖算法 FastCFD [3]的扩展与改进，得到编辑规则挖掘算法 BEFastER. FastCFD 是一种基于深度优先，能挖掘最小的，k-频繁的 CFD 算法.

3.1 算法准备

FastCFD 的目的是对每一个属性 $A\in attr(R)$ 作为规则的右边，找到所有可能对应的规则左边 Y，形成最小的 $\varphi=(Y\to A,t_p)$，其中 $Y\subseteq attr(R)\setminus A$，$\sup(\varphi,r)\geq k$. 我们把这样的规则集记为 Cover($A,r,k$)，显然所有的 k-频繁最小 CFDs 可以由 $\cup_{A\in attr(R)}\mathrm{Cover}(A,r,k)$ 组成. 这样我们的任务主要是计算 Cover(A,r,k). 而差集的覆盖与 Cover(A,r,k) 是有关系的，差集的最小覆盖对应于最小 CFD 的 LHS(φ).

3.1.1 差集

为了计算 Cover(A,r,k)，我们需引入差集的概

念[11], 定义在一对元组 t_1,t_2 的差集表示为:

$$D(t_1,t_2;r)=\{B\in attr(R)\mid t_1[B]\neq t_2[B]\}$$

即元组 t_1,t_2 在属性值上不相等的属性集. 我们定义在 r 上的差集为 $D(r)=\{D(t_1,t_2;r)\mid t_1,t_2\in r\}$. 由于我们锁定了规则右边故只关注 LHS(φ) , 定义 $D_A(r)=\{Y\setminus A\mid Y\in D(r),A\in Y\}$ 为那些包含属性 A 的属性集 Y 除去属性 A 后的属性集. 如果说不存在差集 $Y'\in D_A(r)$, $Y'\subseteq Y$ 则差集 Y 为最小的. 例如, 对表 1 中的数据, 我们可以得到 $D(t_1,t_2)=\{NM\}$,

$D(r)=\{\{NM\},\{PN,NM,STR\},\{CC,PN,NM,STR,ZIP\},$
$\{AC,PN,NM,STR,CT,ZIP\},\{AC,NM,STR,CT,ZIP\},$
$\{CC,AC,PN,NM,STR,CT,ZIP\},\{AC,NM,STR,CT\},$
$\{CC,PN,NM,STR,CT,ZIP\},\{PN,NM\},\{AC,STR,CT,ZIP\}\}$

$D_{\{CT\}}(r)=\{\{AC,PN,NM,STR,ZIP\},\{AC,NM,STR,ZIP\},$
$\{CC,AC,PN,NM,STR,ZIP\},\{AC,NM,STR\},$
$\{CC,PN,NM,STR,ZIP\},\{AC,STR,ZIP\}\}$

$D^m_{\{CT\}}(r)=\{\{AC,NM,STR\},\{AC,STR,ZIP\},$
$\{CC,PN,NM,STR,ZIP\}\}$

得到最小差集后, 需要找到其最小覆盖. 由此, 我们引入覆盖的概念: 令 $Z\subseteq attr(R)$, $X\subseteq P(attr(R))$, $P(attr(R))$ 为集合 $attr(R)$ 的幂集, 如果对每个 $Y\in X,Y\cap Z\neq\varnothing$, 则 Z 覆盖 X ; 如果不存在 Z 的真子集也能覆盖 X , 则 Z 为最小覆盖. 这样可以得到 $D^m_{\{CT\}}(r)$ 的最小覆盖为:

{{STR},{AC,CC},{AC,PN},{AC,NM},{AC,ZIP}}

3.1.2 验证 CFDs

我们找最小差集的最小覆盖是从模式组 (X,t_p) 开始的, 即从满足模式 t_p 的元组记为 r_{t_p} 中计算出 $D^m_A(r_{t_p})$, 其最小覆盖 $Y\subset attr(R)\setminus X\cup\{A\}$ 与 X 组成最小 CFD $\varphi_{\min}=([X,Y]\to A,(t_p,\_,,,\_\parallel a))$. 对于模式组 (X,t_p) 的获取可以使用文献[12]中 GCGROWTH 算法得到, 这里只考虑开集, 所谓开集指的是在保持支持度不变的情况下, 不存在 $Y\subset X,s_p=t_p[Y]$ 这样的模式组 (Y,s_p) , 则 (X,t_p) 为开集. 对常 CFD 和变 CFD 的验证可以根据如下引理[3]来判断.

引理 1.

1) 对任意的常 CFD $\varphi=(X\to A,(t_p\parallel a))$, $r\models\varphi$ $\sup(\varphi,r)\geq k$ 当且仅当 $|r_{t_p}|\geq k$, $D^m_A(r_{t_p})=\varnothing,\pi_A(r_{t_p})=(a)$ 时成立.

2) 对任意的变 CFD $\varphi=(X\to A,(t_p\parallel\_))$, $r\models\varphi$ $\sup(\varphi,r)\geq k$ 当且仅当 $|r_{t_p}|\geq k$, X 覆盖 $D^m_A(r_{t_p})$ 时成立.

引理 1 告诉我们, 测试一个常 CFD 是否为最小 k -频繁的, 需要满足 $|r_{t_p}|\geq k$, $D^m_A(r_{t_p})=\varnothing$, $\pi_A(r_{t_p})=(a)$ 这三个条件; 为了验证 $([X^c,X^v]\to A,(t^c_p,t^v_p\parallel\_))$ 是否为最小 k -频繁变 CFD, 在满足 $|r_{t_p}|\geq k$ 下, 还需具体满足下面两个条件:

a1. X^v 是 $D^m_A(r_{t^c_p})$ 的最小覆盖即不存在 $Y'\subseteq X^c\mid Y'\mid=\mid X^v\mid-1$ 使得 Y' 覆盖 $D^m_A(r_{t^c_p})$.

a2. 不存在 X^c 的真子集 X' , $|X'|=|X^c|-1$ 使得 $X^v\cup(X^c\setminus X')$ 覆盖 $D^m_A(r_{t^c_p[X']})$.

3.1.3 剪枝策略

剪枝策略告诉我们没有必要考虑所有的 k -频繁开集 (X^c,t^c_p) , 即如果 $([X^c,X^v]\to A,(t^c_p,t^v_p\parallel\_))$ 满足要求, 则所有作为 X^c 的超集 X' , 形成的开集 (X',t'_p) 就不用考虑了. 这样可以提高挖掘 CFD 的效率. 下面的引理[3]告诉我们开集作为 CFD 的常模式部分是充分的.

引理 2. 对于变 CFD $\varphi=(X\to A,(t_p\parallel\_))$, 且 $r\models\varphi$, $\sup(\varphi,r)\geq k$, 如果 φ 是最小的, 则 (X^c,t^c_p) 是 k -频繁开集.

3.2 算法描述

为了挖掘出有效的编辑规则, 挖掘编辑规则的算法采用改进的 FastCFD . FastCFD 的主要两个步骤是 FindCover 和 FindMin . 根据引理 2, FindCover 首先要得到在 r 上的所有 k -频繁开集 $F_{r_k}(r)$, 并按开集大小升序排序, 即优先考虑开集中包含属性少的开集. 为了有效地检索 $F_{r_k}(r)$ 中的元素, 用哈希表存放这些元素. 对 $F_{r_k}(r)$ 中的每个项集 (X,t_p) , 计算 $D^m_A(r_{t_p})$ 的最小覆盖, 在该计算过程中会调用 FindMin . 下面给出 BE FastER 的详细过程.

算法BEFastER :

输入：属性集R, R_m关系实例r, r_m，支持度阈值k，相似度阈值d

输出：最小编辑规则集Σ

```
1. Σ:=∅;
2. F_{r_k}(r) := genFreeSets(R,r,k);/*计算得到所有k-频繁开集*/
3. hash table H:=CreateHash(R,R_m,r,r_m,d);
4. for A ∈ R do
5.     Cover(A,r,k):=FindCover(A,r,k,F_{r_k}(r));
6.     for φ ∈ Cover(A,r,k) do
7.         ψ:=((X,X_m)→(A,A_m),t_p[X]);/*X_m, A_m from hash table*/
8.         Σ:=Σ∪ψ;
9. return Σ;
```

过程 CreateHash(R, R_m, r, r_m, d):

```
1. define a hash table H;
2. for all A ∈ R do
3.     max := d; /*保存最相似的*/
4.     for B ∈ R_m do
5.         D(A,B)=|{v | v ∈ D_R(A), v ∈ D_{R_m}(B)}|/|D_R(A) ∪ D_{R_m}(B)|;
6.         if(D(A,B) ≥ max) then max =: D(A,B); add(A,B) to H;
7. return H;
```

过程 FindCover($A,, r, k, \mathrm{F}_{r_k}(r)$)

```
1. cover(A,r,k) := ∅; /*用于保存所有RHS(φ)为A的规则*/
2. for (X,t_p) ∈ F_{r_k}(r) do
3.     r_{t_p} := {t ∈ r | t ≤ t_p};
4.     D^m_A(r_{t_p}) :=genDiffSets(A, r_{t_p}, X);
5.     >_curr := {B ∈ Z | Z ∈ D^m_A(r_{t_p})}; /*按字母排序的属性集*/
6.     Y := FindMin(A,(X,t_p), D^m_A(r_{t_p}), >_curr);
7.     cover(A,r,k) := cover(A,r,k) ∪ Y;
8. return cover(A,r,k);
```

从过程 FindCover 可以发现, 其中调用了两个方法, 分别为 genDiffSets 和 FindMin . genDiffSets 方法的作用是根据满足模式 t_p 的元组, 设定的支持度阈值和属性 A 得到 $D^m_A(r_{t_p})$. FindMin 方法是个递归函数, 主要按深度优先的方式搜索一个按属性字母顺序生成的枚举树, 从根到某个节点的路径可以形成 D_{curr} 的所有子集, 然后判断子集 Y 是否是 $D^m_A(r_{t_p})$ 的最小覆盖, 在该方法中还用 D_{curr} 保存当前还没被 Y 覆盖的差集, 最后返回所有 X 与最小覆盖 Y 形成的CFD 规则. 下面详细介绍这两种方法.

方法 genDiffSets(A, r_{t_p}, X):

```
1. r_{t_p} := ∅; D_A(r_{t_p}) := ∅; ag(r_{t_p}) := ∅;
2. for C ∈ attr(R)\X do
3.     compute π_C := {[u]_C | u ∈ r_{t_p}};
4.     π_C:={c ∈ π_C | |c|>1}; r_{t_p} := π_C ∪ r_{t_p};
5. MC := Max ⊆ {c ∈ π̃ | π̃ ∈ r_{t_p}} /*若c1,c2 ∈ r_{t_p}, c1 ⊂ c2, 则只保留c2*/
6. for π̃ ∈ MC do
7.     for t_i ∈ π̃ do
8.         for t_j ∈ π̃, j > i do
9.             ag(r_{t_p}) := ag(r_{t_p}) ∪ A(t_i, t_j); /*相同的属性*/
10. for X' ∈ ag(r_{t_p}) do
11.     if(A ∈ R − X')
12.         D_A(r_{t_p}) := R − X' − A ∪ D_A(r_{t_p});
13. for Y, Y' ∈ D_A(r_{t_p}) do
14.     if(Y ⊂ Y') then remove Y' from D_A(r_{t_p})
15. return D_A(r_{t_p});
```

上面生成差集的方法是基于等价类划分的, 只保留等价类中元素个数大于 1 的等价类, 且尽量取大集合等价类. 第 3 行计算所有属性在 r_{t_p} 中的划分; 第 4 行合并所有等价类大小大于 1 的等价类; 第 6-9 行求 MC 中所有等价类中所有元组具有相同值的属性集; 与文献[3]中的计算差集的方法相比, 减少了元组对的比较次数. 下面介绍 FindMin .

Method FindMin($A,(X,t_p), D_{curr}, Y, >_{curr}$)

```
1.  CFD_min := ∅;
2.  if >_curr = ∅ but D_curr ≠ ∅ then
3.      return;/*此时不会存在有效的CFD*/
4.  if D_curr = ∅ but >_curr ≠ ∅ then
5.      return;/*此时不会存在有效的CFD*/
6.  if D_curr = ∅ and >_curr = ∅ then
7.      if D^m_A(r_{t_p})=∅, then
8.          if π_A(r_{t_p})={a} or {{a},{b}} then /*r_{t_p}在属性A上的取值只能
                唯一为a或者只允许一个异常值*/
9.              CFD_min:=CFD_min ∪ (X → A,(t_p ‖ a));
10.         else return;
11.     if D^m_A(r_{t_p}) ≠ ∅ then
12.         if (no subset of size |Y|-1 of |Y| covers D^m_A(r_{t_p}) and
13.         no subset X' ⊂ X of |X|−1 Y ∪ (X \ X') covers D^m_A(r_{t_p[X']}))
14.             CFD_min:=CFD_min ∪ ([X,Y] → A,(t_p,_,...,_ ‖ _));
15.         else return;
16. for attributes B ∈ >_curr in order do
17.     D_next := D_curr not covered by B;
18.     >_next := {B̂ ∈ Z|Z ∈ D_next, B < B̂}/*并按字母排序*/
19.     FindMin(A,(X,t_p), D_next, Y ∪ B, >_next);
20. return CFD_min;
```

算法 BEFastER 首先计算了两个全局变量, 一个存放属性对的哈希表, 将 CFD 规则中的属性与主数据中的属性关联起来; 另一个为存放所有 k -频繁开集 $F_{r_k}(r)$ 的哈希表, 并作为 FindCover 的参数.

3.3 算法应用

为了验证算法的有效性, 将其应用于表 1 和表 2, 过程如下:

(1) 根据文献[12]提供的算法可以得到所有的 3-

频繁开集为:

$F_{r_3}(r)$={(CC,01),(CC,44),(AC,131),(PN,2222222),(ZIP,07974),(CT,MH),(AC,908),([CC,AC],[01,908]),([CC,CT],[01,MH])}

(2) 从 F_{r_3} 中取一个开集 (CC,01) 为例, 展开下面的计算.

(3) 扫描表 1 中的数据, 找出满足 t[CC] = 01 的元组有 $r_{CC=01}=\{t_1,t_2,t_3,t_4,t_8\}$.

(4) 调用 genDiffSets 方法, 计算每个属性的划分取等价类中元素大于 1 的等价类, 得到如下结果:

$\pi_{AC}=\{\{t_1,t_2,t_4\}\}$ $\pi_{PN}=\{\{t_1,t_2\},\{t_3,t_4,t_8\}\}$

$\pi_{STR}=\{\{t_1,t_2\}\}$ $\pi_{CT}=\{\{t_1,t_2,t_4\}\}$

$\pi_{ZIP}=\{\{t_1,t_2,t_4\},\{t_3,t_8\}\}$

$MC=\{\{t_1,t_2,t_4\},\{t_3,t_4,t_8\}\}$

这里不妨把规则右边定为 STR, NM 在每个元组中都不一样, 这里不考虑该属性则 $D^m_{STR}(r_{cc=01})$={[PN],[AC,CT]}.

(5) 找 $D^m_{STR}(r_{cc=01})$ 的最小覆盖步骤如图 1 所示.

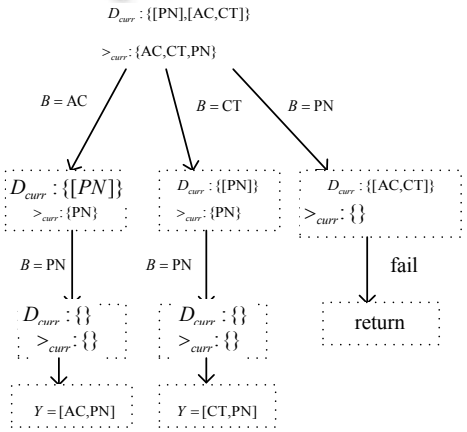

图 1 方法的部分过程

(6) 根据 (5) 中得到 Y, 可以得到两个最小变 CFDs, 分别为:

$\phi'=([CC,AC,PN]\rightarrow STR,(01,\_,\_\|\_))$

$\phi''=([CC,CT,PN]\rightarrow STR,(01,\_,\_\|\_))$

(7) 由于表 1 中存在脏数据且表 1 和表 2 的记录分别只是模式表 R 和 R_m 的部分, 这里我们设定相似度阈值为 0.5, 存在于哈希表有 (CC,CC), (AC,AC), (CT,CT), (PN,TEL), (STR,POST).

(8) 最后得到的编辑规则为:

$\psi'=(([CC,AC,PN],[CC,AC,TEL])\rightarrow(STR,POST),(01,\_,\_))$

$\psi''=(([CC,CT,PN],[CC,CT,TEL])\rightarrow(STR,POST),(01,\_,\_))$

3.4 算法复杂度分析

BEFastER 算法的时间主要消耗在 genDiffSets 方法和 FindMin 上, genDiffSets 中最耗时的是计算等价类所花的时间, 在最坏的情况下, 由关系属性数目和满足模式 t_p 的元组数决定, 故此方法的时间复杂度为 $O(|R||r_{t_p}|^2)$; FindMin 是个递归方法, 会占用一定的栈空间, 但可以减少遍历 $R\setminus\{A\}$ 所有子集的时间, 栈空间由递归次数决定, 递归次数越多, 程序结束越迟, 相应的时间开销就越大, 在最坏情况下, 时间复杂度为 $O(|>_{curr}|\log(|>_{curr}|))$, $>_{curr}$ 为最开始传入的 $>_{curr}$. 对比 genDiffSets 所花的时间, 这点时间可以忽略. 如果不考虑 GCGROWTH 计算开集的时间, 则 BEFastER 的总的时间开销为 $O(|F_{r_k}(r)||R||r_{t_p}|^2)$, $|F_{r_k}(r)|$ 为开集数目.

4 实验

本节主要通过来自医院的真实数据 HOSP[13]来进行实验研究, 将从响应时间和挖掘的规则数目两个指标来验证算法的有效性. 并比较了改进算法与扩展的文献[3]算法. 主要改进在计算差集的方法上和判断常 CFDs 的条件上.

4.1 实验数据和环境

实验数据: 由于主数据的获取比较难, 假设以处理的 HOSP 为主数据 HOSPM, 即去除 HOSP 中的空值和异常值记录, 得到干净的主数据进行模拟. 这样一来匹配主数据属性这步基本可以忽略. HOSP 包括七个属性和 9455 条记录.

实验环境: 我们用 java 语言在 window7 操作系统, Mysql 数据库管理系统, Intel Core i5-2400 3.10GHz CPU 和 4G 内存下实现了本文的算法. 每次实验重复了 5 次, 取平均结果.

4.2 实验结果

在数据集 HOSP 和 HOSPM 下, 随着支持度阈值的变化, 本文改进的算法 BEFastER 和基于文献[3]的 FastER 的响应时间如图 2 所示.

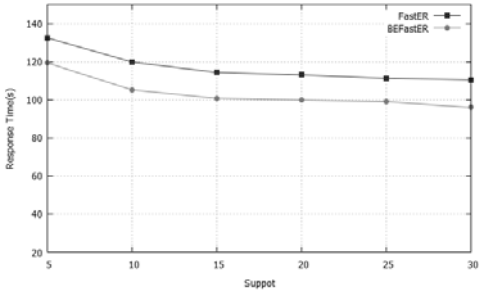

图 2 关于支持度阈值 k 的时间变化

从上面的结果可以知道, 两种算法基本对支持度的敏感程度都不大, 和文献[3]的结论一致, 但在所得规则数一样的情况下, 我们改进后的算法比直接扩展的性能更好.

另一个实验结果是 BEFastER 随支持度的变化, 挖掘的规则数的变化情况, 如下图 3 所示, 规则数随支持度的加大, 不管是常规则数还是变规则数都在明显地减少.

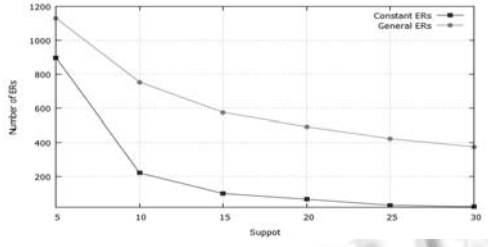

图 3　关于支持度阈值 k 的规则数变化

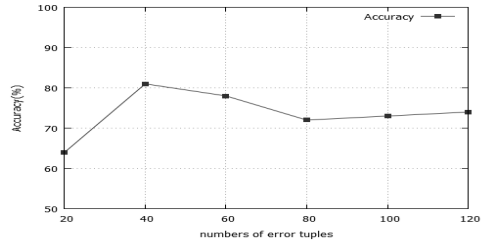

图4 数据集 D 中包含错误元组数对修复正确率的影响

为了验证所得的编辑规则是否能修复数据, 我们根据编辑规则语义进行了修复实验, 将支持度为 20 得到的编辑规则应用于添加了错误的数据集 HOSP. 由图 4 的结果可以知道, 修复准确率能稳定达到 70%-80%, 不能达到 100%的主要原因是我们不能确定 t[LHS(φ)] 的值是否正确, 如果待修复的元组中错误的 t[LHS(φ)] 匹配了 t_m[LHS(φ)] , 那么将会导致错误的修复. 另一个原因是在存在错误样本情况下, 我们挖掘的编辑规则很难做到完全正确.

5　结语

本文提出了基于条件函数依赖挖掘的编辑规则挖掘算法, 从编辑规则的形式化定义来看, 与 CFD 相比, 只是多了主数据中的属性, 所以本文的第一个工作是重点挖掘最小 CFDs 包括常 CFDs 和变 CFDs, 此工作的主要耗时任务是计算差集和最小覆盖, 而本文计算差集的方法大大减少了元组对的比较, 在得到差集为空时, 考虑到输入样本中存在错误, 我们降低了值非得唯一的要求, 从而可以得到更多的常 CFDs. 对得到的 CFDs 只利用其模式组的左边部分, 用于匹配待修复的元组; 第二个工作是在主数据中找对应的属性, 由于来自不同数据源的表结构存在差异, 我们采用了基于定义域的简单计算模型, 理想状态下对应属性定义域应该是一样的. 但由于输入样本中存在脏数据, 定义域只能近似匹配. 本文通过实验验证了我们所提算法的有效性, 能在有限的时间里挖掘出编辑规则, 并对所得的编辑规则按照编辑规则语义进行了质量评估, 进一步验证了我们的算法确实实用有效.

参考文献

1 Fan W, Li J, Ma S, et al. Towards certain fixes with editing rules and master data. Proc. of the VLDB Endowment, 2010, 3(1-2): 173–184.

2 Cong G, Fan W, Geerts F, et al. Improving data quality: Consistency and accuracy. Proc. of the 33rd international conference on very large data bases. VLDB Endowment. 2007. 315–326.

3 Fan W, Geerts F, Li J, et al. Discovering conditional functional dependencies. IEEE Trans. on Knowledge and Data Engineering, 2011, 23(5): 683–698.

4 Fan W, Geerts F, Jia X, et al. Conditional functional dependencies for capturing data inconsistencies. ACM Trans. on Database Systems (TODS), 2008, 33(2): 6.

5 胡艳丽,张维明.条件依赖理论及其应用展望.计算机科学, 2009,36(12):115–118.

6 Fan W, Ma S, Tang N, et al. Interaction between record matching and data repairing. Journal of Data and Information Quality (JDIQ), 2014, 4(4): 16.

7 Huhtala Y, Kärkkäin J, Porkka P, et al. TANE: An efficient algorithm for discovering functional and approximate dependencies. The computer journal, 1999, 42(2): 100–111.

8 刘波,耿寅融.数据质量检测规则挖掘方法.模式识别与人工智能,2012,25(5):835–844.

9 Medina R, Nourine L. A unified hierarchy for functional dependencies, conditional functional dependencies and association rules. Lecture Notes in Computer Science, 2009, 5548: 98–113.

10 Chiang F, Miller R J. Discovering data quality rules. Proc. of the VLDB Endowment, 2008, 1(1): 1166–1177.

11 Wyss C, Giannella C, Robertson E. Fastfds: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances extended abstract. Data Warehousing and Knowledge Discovery. Springer Berlin Heidelberg. 2001. 101–110.

12 Li H, Li J, Wong L, et al. Relative Risk and Odds Ratio: A Data Mining Perspective (Corrected Version). Pods, 2005: 368–377.

13 http://www.hospitalcompare.hhs.gov/.