DOI: 10.13656/j.cnki.gxkx.2005.02.011
广西科学 Guangxi Sciences 2005, 12(2): 118~122

数据清洗前的预处理方法*
Pre-Processing for Data Cleansing

唐懿芳[1],钟达夫[1],张师超[1,2]
Tang Yifang[1], Zhong Dafu[1], Zhang Shichao[1,2]

(1. 广西师范大学数学与计算机科学学院,广西桂林 541004; 2. 悉尼理工大学信息技术学院,澳大利亚悉尼)
(1. Coll. of Math.& Comp. Sci., Guangxi Normal Univ., Guilin, Guangxi, 541004, China;
2. Faculty of Info. Tech., Sydney Tech. Univ., Sydney, Australia)

摘要:为提高数据清洗的质量,提出消除脏数据域、使用统一的缩写、数据的转换等预处理方法,基于这3种方法和链表存储复制记录算法,设计一个数据清洗的系统,与其他方法的效率与准确程度比较可知,该系统的数据准确程度要高于现有的数据清洗系统.

关键词:数据清洗 脏数据 预处理 外部源文件
中图法分类号: TP311.131 文献标识码: A 文章编号: 1005-9164(2005)02-0118-05

Abstract For improving the quality of data cleaning, it provides three pre-process methods, such as eliminating dirty data, using unified abbreviation, data converstion. Based on these methods, using link table to store replicate recorders algorithm, implementing a data cleansing system. This cleaning system has a higher veracity than the existing one.

Key words data cleansing, dirty data, pre-processing, external source file

当前的企事业单位都面临着处理海量数据的挑战.这些大型的数据库通常由于某些原因而包含着数据错误或是不一致等现象,引起错误的原因有[1]: (1)错误的输入而导致不正确的数据值; (2)因为输入格式的不同或是使用不同的缩写形式而引起的不一致的数据; (3)不能完全搜集数据的信息而导致丢失的数据.所有这些原因都可能使企事业单位在做出重大决策时,不可避免地出现偏差,从而造成巨大的损失.

要避免这种情况,就要解决所谓的"garbage in, garbage out"[2]的问题.数据清洗的处理正是为了解决大型数据库中常有的输入错误、不一致等现象而提出,但当前大部分数据清洗的系统都忽略了数据预处理的操作.针对这一情况,本文提出了在数据清洗操作前进行一些简单的预处理,这些预处理提高了数据清洗的质量.

进行预处理主要有3个步骤: (1)清除脏数据域.通过一些外部函数和外部源文件纠正数据记录的一些简单的错误,其目的是去除数据的输入错误. (2)使用统一的缩写.根据缩写表达与全称的对应关系,对所有数据进行一个标准化的处理. (3)数据的转换.主要对一些表示格式不同的数据进行转换.这个转换过程就是要把这些表示不一致的数据转换成表示一致的数据,且还能按照一定的要求把一个数据表转换成多个不同结构的数据表.

1 清除脏数据域

现有的数据清洗技术中比较常用的排序邻居方法和复制记录删除方法的主要思想都是先从数据记录中抽取出具有代表性的键值,然后依据这个键值对整个数据库进行排序,有一个尺寸大小固定的窗口在这些排好序的数据记录间移动,窗口内的数据记录依次进行两两比较[3].这种方法在理想情况下,表示同一实体的复制记录是相邻的,因而能在这个窗口中找到.这个方法的关键在于选取键值的优劣.但如果这些数据本身存在错误,而键值又是从这些包含错误的数据中直接抽取出来的,则用于排序的键值也有可能包含错误,一些表示同一实体的复制记录将不能在窗

收稿日期: 2005-01-06
修回日期: 2005-03-07
作者简介: 唐懿芳(1976-),女,广西富川人,讲师,主要从事计算机网络和数据处理研究.
* 澳大利亚国家大型项目(ARG DP0343109)资助.

口中检测出来,从而不能保证这些技术的有效性.而清洗预处理中的清除脏数据步骤可以去除一些简单的数据错误,使从数据中抽取的键值更为准确.

脏数据的存在主要有几种原因: (1)在大型的数据库中,经常存在输入错误和打印错误; (2)输入错误将导致不正确的数据和丢失数据的情况; (3)为了加快数据的输入,很多数据库采用简写的方式,而同一实体不同的简称将产生不一致的数据; (4)不同的输入格式也是造成脏数据的另一重要原因.此外还有其它可能造成脏数据的原因,这里就不再一一详述.本文试图清除这些脏数据,并提出一些新方法使缩写标准化.

设想有一个如表 1所示的学生记录数据库 STU,由于脏数据的存在,这些记录中可能存在有相同的记录,我们称之为复制记录.同一数据库中是不允许出现复制记录的,为了更有效地找到这些复制记录,必须首先清除这些脏数据.我们利用外部源文件验证这些记录数据,并解决数据表示之间的冲突.

表 **1** 含有脏数据的数据记录

Table 1 Data records included dirty data

学号 Student number	姓名 Name	性别 Sex	地 址 Address
940101	胡汉成 Huhancheng	男 Male	合浦师范 Hepu normal academy
940102	何雪莲 Hexuelian	F	苍梧县高中 Cangwu county high school
940103	吴承为 Wuchengwei	M	合山市初级中学 Heshan city preliminary school
940104	刘祖华 Liuzuhua	男 Male	河池地区高中 Hechi regional high school

外部源文件主要用于查询数据,它可以做成关系数据库的形式.表 2是一个外部源文件的例子.这些外部源文件应该是由一些比较权威的机构提供,包含着比较准确比较完整的信息.表 2中作为实体的特定的人仅用一条记录表示其信息.本文的主要思想就是利用这个外部文件来清除脏数据.我们注意到表 2存在着一个函数依赖关系: 学号→姓名,性别,地址.学号作为记录的键值,具有唯一表示实体的性质.清除脏数据的处理将依据外部文件的键值进行.在清除脏数据的处理过程中,系统首先根据学号在外部文件中查找出相应的记录,然后将包含脏数据的记录与外部文件的记录做比较,若发现错误的或不同表示的数据,则进行修改.

键值也可能包含错误,表 2中的键值—— 学号如果存在错误无外乎有 2种情况:

(1)错误的学号在外部文件中查不到.遇到这种情况把这个记录直接提交给用户.

(2)错误的学号是另一个学生的学号.

表 **2** 外部源文件

Table 2 External source file

学号 Student number	姓名 Name	性别 Sex	地 址 Address
940101	吴汉成 Wuhancheng	男 Male	合浦师范 Hepu normal academy
940102	何雪莲 Hexuelian	女 Female	苍梧县高中 Cangwu county high school
940103	吴承为 Wuchengwei	男 Male	合山市初级中学 Heshan city preliminary school
940104	赖祝兴 Lanzhuxing	男 Male	阳朔县高中 Yangshuo county high school
940105	刘祖华 Liuzuhua	男 Male	河池地区高中 Hechi regional high school

表 2中“ 940104刘祖华”的记录就属于这种情况.系统处理这种情况时计算数据库记录与外部源文件记录的相似程度.

本文设计了一种基于领域权值计算相似度的方法来计算数据库记录和外部文件记录的相似程度.如果记录的相似程度超过一个设定的阈值,就认定数据库的记录与外部文件的记录表示同一个实体,然后自动依据外部文件改成正确的记录.如果相似程度只略微低于阈值,则提交给用户,由用户决定是否表示同一实体.

下面将介绍这种基于领域权值的相似度计算方法.领域权值的大小显示了计算记录间相似程度时这个领域的相对重要程度.一般有代表性的域有较高的权值,比如本节的例子姓名的权值就应该比性别的权值要大.各领域权值的大小由用户或专家提供,所有域的权值总和应为 1.在计算记录相似度的时候,首先把每一个领域值看做是一个字符串,并使用编辑距离 editdist[4]计算记录中领域的相似程度.所谓编辑距离就是一个字符串 *string*1转化成另一个字符串 *string*2所要删除、修改、插入的字符数.假设 *string*1的长度为 m,相似度 $similar = 1 - editdist/m$.

定义 **1** 领域相似度计算.

假设记录 X的一个领域包含子领域 $S_{x1}, S_{x2}, \cdots, S_{xn}$,对应的记录 Y有子领域 $S_{y1}, S_{y2}, \cdots, S_{ym}$.每个子领域 S_{xi}, $1 \leqslant i \leqslant n$与 S_{yj}, $1 \leqslant j \leqslant m$进行比较,让 $MS_{x1}, MS_{x2}, \cdots, MS_{xn}, MS_{y1}, MS_{y2}, \cdots, MS_{yn}$ 分别表示各自子领域与其它子领域相比较的最大相似度,记

录 X 与记录 Y 对应的领域相似度 $Sim_F(X,Y)$ 为:

$$Sim_F(X,Y)=\frac{\sum_{i=1}^{n}MS_{xi}+\sum_{i=1}^{m}MS_{yi}}{n+m}. \quad (1)$$

定义 **2** 记录相似度计算.

假设数据库有领域 $F_1,F_2,\cdots,F_n$,它们的权值分别为 $W_1,W_2,\cdots,W_n$,$Sim_{F1}(X,Y),\cdots,Sim_{Fn}(X,Y)$ 是已经计算得到的领域相似度,则记录 X 与记录 Y 的相似度 $Sim(X,Y)$:

$$Sim(X,Y)=\sum_{i=1}^{n}Sim^{Fi}(X,Y)\times W_i. \quad (2)$$

根据这些计算和分析,可以把表 1的脏数据进行清除,清洗后的数据如表 3所示.

表 **3** 清洗处理后的数据

Table 3 Cleansing data being cleaned

学号 Student number	姓名 Name	性别 Sex	地址 Adress
940101	吴汉成 Wuhancheng	男 Male	合浦师范 Hepu normal academy
940102	何雪莲 Hexuelian	女 Female	苍梧县高中 Cangwu county high school
940103	吴承为 Wuchengwei	男 Male	吾山市初级中学 Heshan city preliminary school
940105	刘祖华 Liuzuhua	男 Male	河池地区高中 Hechi regional high school

2 缩写标准化处理

输入简写的目的是为了加快输入速度,但简写有可能造成不一致的数据,如“广西师范大学”有时简称“广西师大”,在不经处理的情况下,系统检测这两个名称时认为是不同的,这就是由缩写导致的数据不一致现象.解决这个问题关键就是在记录中要么都采用全称,要么都使用一致的缩写形式.缩写在中文数据库中一般有 3种情况:

(1)缩写是全称的前缀和中间一些字的组合,比如上面提到的“广西师范大学”简称“广西师大”;

(2)缩写是全称的前缀和后缀,如“中华人民共和国”简称“中国”;

(3)缩写是一些地名的简称,比如“广西”也称为“桂”.

英语中人物的名称存在着多样性,同一个人的名称拼写可以有很多种不同的形式,各个机构都是按照自己的习惯表示对象.“John Smith”、“Smith John”和“J. Smith”在拼写上有一些轻微的不同,但实际上它们可以表示同一个人.合并数据库的时候,如果没有把它们正确地识别出来,同样会造成不一致或者重复的数据.除此以外,英文缩写还有下列一些情况[5]:

(1)缩写是全称的前缀,比如“Univ”是“University”的简写;

(2)缩写联合了全称的前缀和后缀,比如“Dept”和“Department”;

(3)缩写是开首的几个字母,如“UCSD”和“University of California, San Diego”;

(4)最后一种情况是把一些短语中词的前缀联合起来表示这些词,如“Caltech”的全称是“California Institute of Technology”.

根据这几种情况,系统在检测它们相似性的时候,首先把一些对应的子领域提取出来,然后利用数据字典检查对应的缩写,同时可以运用 TFIDF方法[6]检查缩写词在全称中的权重,以此得到它们的相似度.这里我们仅提出这个解决缩写问题的新思路,数据字典的建立和算法的实现和完善有待于将来工作的继续开展.

3 数据转换

数据转换处理一方面是调用一些外部函数简化数据库属性的值,使属性值的特征更突出,或是使属性值标准化,另一方面是把要合并的数据库之间的模式转换成一致的模式,从而使数据记录之间更深层的比较成为可能.因为我们处理的对象是大型数据库,现代的数据库一般都是关系型数据库,而 SQL语言是处理关系数据库最有效的工具之一.

本文提出一个利用改进的 SQL指令性语言完成对数据转换的新方法.这种语言既具有 SQL语言对付大型数据库查询的效率,同时具有较好的表达能力和灵活性.较强的表达能力表现在它能很好地表示不同类型的数据转换操作,灵活性表现在它能调用不同的外部函数进行不同的操作.数据转换的输入数据是一些可能含有错误的数据或格式不同的关系数据库.有效地执行这种 SQL语言需要一个 SQL查询引擎和外部函数执行引擎.外部函数执行引擎实际上就是一个外部函数库,它的建立将是我们将来针对数据转换的主要工作之一.

本文仍然用表 2的学生数据库 STU为例,如果要把地址的省、市、县、学校这些子领域分开来,数据转换可以写成:

```
CREATE TRANSM1
SELECT s.stuid, s.name, s.sex, s.city,
s.school
FROM STU s
  LET [city, school] = ExtractAddress
(s.address)
      s.sex = if(s.sex = = "M") THEN
RETURN"男"
      ELSE RETURN "女"
```

这个例子把地址领域分解成更小的子领域(city, school)并把性别的缩写转换成一致的格式. LET从句是一个如何进行转换操作的说明性从句,它的形式一般是一系列的变量说明: <var_name> = <expression>.变量的说明由一些表达式定义,用以对 FROM从句关系表的各记录变量进行定义,这些表达式可以调用由系统平台定义的外部函数,利用外部函数执行引擎调用外部函数库.而 SELECT从句把 LET从句中的所得的变量值投影到 FROM中的关系表中.

一个数据转换可以同时从一个输入表中生成多个关系表,这时需要定义多个 SELECT从句,其中 INTO <table> 分别详细地陈述输出的多个表.假设上面的学生记录表 STU,我们想生成 2个关系表,一个是标准化处理后的关系表 NORM- STU,把原来数据库中的性别表示标准化,把地址属性分解为具体的城市名称和学校名称,另一个是学生的学号对应表 STUNUM,包含学生的学号和姓名两个属性.这种情况下,命名为 M1的数据转换并不具体表示输出的关系表,而由 INTO从句说明,这个数据转换可表示为:

```
CREATE TRANS M1
  SELECT s.stuid, s.name, s.sex, s.city, s.school INTO NORM-STU
  SELECT s.stuid, s.name INTO STUNUM
  FROM STU s
    LET [city, school] = ExtractAddress(s.address)
    s.sex = if(s.sex = = "M") THEN RETURN "男"
    ELSE RETURN "女"
```

4 基于预处理技术的数据清洗系统

本文用 Visual C++ 6.0语言编写了一个包含预处理技术的数据清洗系统,并在 CPU为 Intel赛扬 1.7GHz,内存为 256MB的机器上运行,所用的操作系统为 Windows2000.此清洗系统主要是处理在同一个数据源中存在多个记录表示同一实体的复制记录的问题,我们在数据清洗前附加了一些预处理技术,此系统采用链表存储复制记录.本文实验所用的数据为学生记录数据,记录属性包括学生学号、姓名、性别、出生年月、邮政编码、家庭地址、联系电话共 7个属性.实验共输入学生记录 875个,通过一些复制处理,并运用随机错误处理程序,得到学生记录数为 2412个.人工计算得到包含 2个记录数以上的聚类为 218个,其中最大的聚类包含 14个记录.

识别复制记录算法的准确性指的是,我们找到的复制记录表示的确实都是同一记录,即不存在假的正例.而算法效率指的是,我们应该尽可能地把所有的复制记录都找到,其中文献[7]还用到信息检索中的出错率和回召率作为衡量查找复制记录算法的标准.

定义 3[7] 回召率指的是复制记录被正确识别的百分比,它能衡量算法的效率.假设有 5个记录 A_1, A_2, B_1, B_2, C_1,其中 $\{A_1, A_2\}$, $\{B_1, B_2\}$ 分别都表示同一实体,是复制记录,但运用算法检测的结果 $\{A_1, C_1\}$, $\{B_1, B_2\}$ 是复制记录.则这个算法的回召率就是 3/4× 100% = 75% .

定义 4[7] 出错率指错误识别复制记录占总复制记录数目的百分比.计算得到上例中的出错率为 1/4 × 100% = 25% .

因为错误识别的复制记录更容易被判定,所以选择出错率来验证算法的正确性.如,上面例子中的出错率为 1/4× 100% = 25% .

学生记录表的家庭地址是一个复合属性,应用本文提到的数据转换方法,把地址的城市、县,具体街道、工作单位,以及门牌号码分解成子属性,同时利用外部源文件进行数据清洗前的预处理操作,针对家庭地址的城市与邮政编码是否对应这一问题,可以清除一部分脏数据.其外部源文件是根据邮政局发行的邮政代码表而建立,我们的实验数据范围很小,只涉及广西,所以在建立外部源文件时,要达到百分百的准确率相对比较容易.表 4给出了对经过预处理和没经过预处理的检测复制记录方法进行比较的实验结果,检测复制记录运用当前用得较多的排序邻居方法[3]

表 4 不同方法的效率和准确程度比较

Table 4 Efficient and correct comparing in different methods

清洗方法 Used method	回召率 Call	出错率 Error
未经过预处理的排序邻居方法 (k = 16) Sorted-neighbor method before data pre-processing (k = 16)	0.970	0.165
未经过预处理的排序邻居方法 (k = 8) Sorted-neighbor method before data pre-processing (k = 8)	0.94	0.24
经过预处理的排序邻居方法 (k = 16) Sorted-neighbor method after data pre-processing (k = 16)	0.972	0.11
经过预处理的排序邻居方法 (k = 8) Sorted-neighbor method after data pre-processing (k = 8)	0.965	0.20
未经过预处理的 Canopy聚类检测复制记录方法 Canopy clustering method to detect duplicate record before data pre-processing	0.966	0.18
经过预处理的 Canopy聚类检测复制记录方法 Canopy clustering method to detect duplicate record after data pre-processing	0.976	0.11

和 Canopy技术的检测复制记录方法[8],其中排序邻居方法的窗口尺寸本文选择了 2种情况做比较: k = 16或 k = 8.

从表 4可以看出,经过预处理的复制记录检测方法的准确率要高于未经过预处理的复制记录检测方法,但由于所用的实验数据不大,所以未能充分显示出经过预处理的优越性.

5 结束语

本文利用外部源文件清除脏数据和标准化简写,保证了数据库中名称的一致性.在用外部源文件清除脏数据的过程中,使用领域权重计算来记录的相似性.本文还提出一种利用类似 SQL的语言对数据库转换成所需模式的新思路,该思路新颖、实用.在将来的工作中,我们将建立相关的数据字典,并把算法运用到大型的数据库中,以进一步检验该方法的优越性.

参考文献:

[1] Ktmball R. Dealing with dirty data[J]. DBMS, 1996, 9(10): 55-57.

[2] Lee M L, Lu H, Ling T W, et al. Cleansing data for mining and warehousing [A]. In: Proceedings of the 10th International Conference on Database and Expert Systems Applications, 1999. 751-760.

[3] Hernandez M, Stolfo S. The merge/purge problem for large databases [A]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. 1995, 127-138.

[4] Levenshtein V. Binary codes capable of correcting deletions [J]. Insertions and Reversals. Soviet Physics-Doklady10. 1966, 10: 707-710.

[5] Monge A, Elkan C. The field-matching problem: algorithm and applications [A]. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. 1996.

[6] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval[J]. Information Processing and Management, 1988, 24(5): 513-523.

[7] Lee M L, Ling T W, Low W L. IntelliClean a knowledge-based intelligent data cleaner[A]. In: Proceeding of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. 2000, 290-294.

[8] McCallum A, Nigam K, Ungar L. Efficient clustering of high-dimensional data sets with application to reference matching[A]. In: Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining, 2000. 169-178.

(责任编辑: 黎贞崇)

(上接第 117页 Continue from page 117)

表等方法入手.

6 结束语

本文对事务数据库的数据特性做了深入的研究,指出了: $|I|$与 $supp(i)$是相对稳定的,事务项 i发生的次数及 $|DB|$在事务数据库中近似线性增长,$minsupp$直接决定频繁项集数量的多少,对挖掘结果有决定性作用.在事务数据库特性的基础上分析了 Apriori算法的复杂性,提出访问产生低阶频繁项集时访问 C_k及 L_k复杂性更高的新观点,进一步阐述有效控制 $|C_k|$及 $|L_k|$的必要性, Apriori算法对挖掘长频繁项集存在访问冗余性,当然 DB进行预处理也应为一种有效途径.

参考文献:

[1] Chengqi Zhang, Shichao Zhang. Association rule mining modelandalgorithms[J]. SpringerI, 2002, 2307: 33-39.

[2] Wu Xindong, Chengqi Zhang, Shichao Zhang. Mining both positive and negative association rules [C]. Proceedings of 19th International Conference on Machine Learning, Sydney: Australia, 2002. 658-665.

[3] GIWebb. Efficientsearchforassociationrulesproc[C]. ACM SIGKDD Int l Conf. Knowledge Discovery and Data Mining, 2000: 99~ 107.

[4] 李绪成,王保保.挖掘关联规则中 Apriori算法的一种改进[J].计算机工程, 2002, 28(7): 104-105.

[5] 陆丽娜,陈亚萍,魏恒义,等.挖掘关联规则中的 Apriori算法的研究[J].小型微型计算机系统, 2000, 21(9): 940-943.

[6] Zhang C, Zhang S. Collecting quality data for database mining[C]. Proceedings of the 14th Australian Joint Conference on Artificial Intelligence, 2001, 593-556.

[7] Zhang S, Zhang C. Mining small database by collecting knowledge[C]. Proceedings of DASFAA01, 2001, 174-175.

[8] Wu X, Zhang S. Synthesizing high-frequency rules from different data sources [J]. IEE Transaction on Knowledge and Data Engineering, 2003, (15) 2: 353-367.

[9] Lee G, Lee K L, Chen A L P. Efficient graph-based algorithms for discovering and maintaining association rules in large databases[J]. Knowledge and Information Systems, 2001, (3): 338-335.

[10] 何炎祥,石 莉,张 戈,等.关联规则的几种开采算法及其比较分析[J].小型微型计算机系统, 2001, 22(9): 1065-1068.

(责任编辑: 黎贞崇)