计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20148218
53(3):559-570,2016

大数据环境下高维数据的快速重复检测方法

朱蔚恒[1] 印 鉴[2] 邓玉辉[1] 龙 舜[1] 邱诗定[1]
[1](暨南大学信息科学技术学院 广州 510632)
[2](中山大学信息科学与技术学院 广州 510006)
(tzhuwh@jnu.edu.cn)

Efficient Duplicate Detection Approach for High Dimensional Big Data

Zhu Weiheng[1], Yin Jian[2], Deng Yuhui[1], Long Shun[1], and Qiu Shiding[1]
[1](*College of Information Science and Technology, Jinan University, Guangzhou* 510632)
[2](*School of Information Science and Technology, Sun Yat-sen University, Guangzhou* 510006)

Abstract The big data era has huge quantity of heterogeneous data from multiple sources be widely used in various domains. Data from multiple sources and of various structures make data duplication inevitable. In addition, such a large amount of data generates an increasing demand for efficient duplicate detection algorithms. Traditional approaches have difficulties in dealing with high dimensional data in big data scenarios. This paper analyses the deficiency of traditional SNM(sorted neighbour method) methods and proposes a novel approach based on clustering. An efficient indexing mechanism is first created with the help of R-tree, which is a variant of B-tree for multi-dimensional space. The proposed algorithm reduces the comparisons needed by taking advantage of the characteristics of clusters and outperforms existing duplicate detection approaches such as SNM, DCS, and DCS++. Furthermore, based on the apriori property of duplicate detection, we develop a new algorithm which can generate the duplicate candidates in parallel manner of the projection of original dataset and then use them to reduce search space of high-dimensional data. Experimental results show that this parallel approach works efficiently when high-dimensional data is encountered. This significant performance improvement suggests that it is ideal for duplicate detection for high dimensional big data.

Key words big data; high dimension data; data mining; data preprocessing; duplicate detection

摘 要 大数据时代多源、异构、海量的数据正逐渐成为各种应用的主流.多源异构不可避免地会使数据出现重复,同时庞大的数据量对重复检测的效率提出了极高的要求,传统技术在大数据环境下并不能很好地对高维数据进行重复检测,就此问题展开研究,分析了传统 SNM 类方法的不足,将重复问题概化为一类特殊的聚类问题,利用 R-树建立了高效的索引,利用聚类簇的特性减少了在 R-树叶子中比较的次数,利用重复检测的 Apriori 性质实现了对高维数据集并行处理.实验结果表明,提出的算法能有效地提高高维数据的重复检测效率.

收稿日期:2014-11-06;**修回日期**:2015-05-05
基金项目:国家自然科学基金项目(61472453,61272073,61401177,61572232,U1401256,U1501252);广东省自然科学基金项目(S2013020012865);广东省科技计划基金项目(2013B010401017)
This work was supported by the National Natural Science Foundation of China (61472453, 61272073, 61401177, 61572232, U1401256, U1501252), the Natural Science Foundation of Guangdong Province of China (S2013020012865), and the Science and Technology Planning Project of Guangdong Province (2013B010401017).
通信作者:龙舜(tlongshun@jnu.edu.cn)

关键词 大数据;高维数据;数据挖掘;数据预处理;重复检测

中图法分类号 TP391

重复检测(duplicate detection)是指从多源数据集中检查表达相同实体的重复记录.现实生活中许多应用需要整合来自不同数据源的数据.在这些实际任务中,经常会碰到这样的问题:同一个实体在不同数据源中有不同的表达,这些表达之间存在一定的差异,不能直接一一对应,这种现象称之为重复记录.导致重复记录的原因很多,如多重数据结构、名称拼写错误、不通用的别名、不同的缩写、方言表达、不完全匹配的记录以及数据由高精度系统导入低精度系统(从 64 b 系统导入 32 b 系统)等.这些重复记录为统计、数据分析、数据挖掘等任务造成了很大的困扰,并直接影响最终的结果.因此,在许多需要处理来自多个数据源数据的实际应用中,如基于多个电商平台/社交网站数据的市场分析、网络营销等,必须要对这些来自多数据源的数据进行重复检测.

随着以博客、社交网络、基于位置的服务为代表的新型信息发布方式的不断涌现,以及云计算、物联网等技术的兴起,数据正以前所未有的速度在不断地增长和累积,大数据时代已经来到[1].对大数据时代的大部分应用而言,收集数据已不再是发展的瓶颈,而如何保证收集到数据的一致性[2]正日渐成为必须面对的问题.

重复检测问题是一个被广泛研究的问题[3-4],在统计学、人工智能、数据库等领域均对其进行了深入的研究,针对重复检测的各个方面提出了不少有效的方法.然而,随着大数据时代来临,传统问题重新面临新的挑战.大数据时代数据来源广、结构复杂、数量巨大[1],多源数据必然导致重复记录的产生;对大量非结构化和半结构化数据处理也会导致重复记录的增加;庞大的数据量则对重复检测方法的效率提出了更严格的要求.因此大数据环境下的高维数据集对传统的数据预处理,数据分析技术提出更高的要求.

当前对重复检测的研究主要基于如下思想:分块(blocking)以及窗口(windowing),其代表性方法是 Hernandez 和 Stolfo[5]提出的 SNM 方法及其改进.在实践过程中我们发现,当进行相似性判别的维度较多时,此类方法的效率仍不理想.为了更有效地对高维数据进行重复检测,本文借鉴了不确定性的概念,将重复问题的实质视作对象不确定性的现实体现,由此可将重复检测问题看成一类特殊的聚类问题,并据此提出了一种基于空间聚类的高效处理重复检测问题的算法 DDR(duplicate detection using R-tree)和一种利用 DDR 以及高维数据重复检测的 Apriori 性质进行剪枝的高维大数据重复检测方法.实验结果表明,在处理高维数据时与传统的重复检测算法相比 DDR 能节省大量的对比次数,而本文提出的重复检测框架也能有效地处理大数据环境下的高维数据重复检测问题.

1 相关工作

1.1 重复检测研究现状

重复检测是数据预处理的一个重要问题,在不同场景下也被称为实体匹配(entity matching)、对象匹配(object matching)、混合/净化(merge/purge)、记录链接(record linkage)或记录一致(record reconciliation).重复检测问题在数据统计、数据分析和数据挖掘等领域有很多的实际应用[3-4].

例如某从事邮件营销的公司利用爬虫从 2 个电商平台采集到一些特定商品的用户购买记录,经初步清洗后得到如表 1 所示数据集.从表 1 可以看出,虽然 2 个用户信息并不完全一致,但它们非常类似,从邮件营销的角度,将这 2 个用户看作同一个人寄送促销广告是很合理的.类似这样的记录对在数据集中有很多,因此公司往往希望在寄送促销广告之前能通过 ID、生日、性别、收货地址等属性把他们找出来,这就是一个典型的重复检测问题.

Table 1 Information Collected from Various E-Bussiness Companies

表 1 从不同的电商平台收集的信息

ID	Data Source	Name	Date	User ID	Birthday	Gender	Address
001	天猫	Headphone	14.5	Zhuw*	76.9	Man	广州市暨南大学计算机系
002	京东	Mobile	14.7	Zhu*h	1976	Man	广州市黄埔大道西暨南大学计算机系

一般地,对来自不同数据源的数据进行重复检测的流程如图 1 所示[5].来自 2 个不同数据源的数据在经过清洗和标准化之后,利用高效的索引对数据进行初步的归纳整理,然后从若干角度对每一对记录进行比较并得到一个相似度向量度量,最后利用一个有监督的分类器判断这一对记录是否为重复记录.

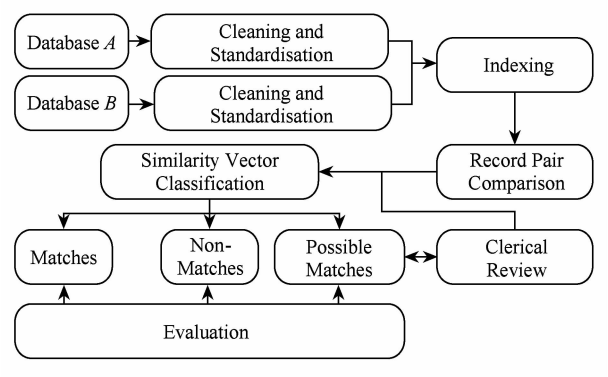


Fig. 1 Duplicate detection for two data sources.

图 1 对 2 个数据源数据进行重复检测的流程

从图 1 中的流程可以看出,决定重复检测效率的关键是如何利用索引来减小记录对的比较次数.近年来,相关领域的主要工作有:韩京宇等人[6]对大数据量的相似记录检测问题展开了研究,针对字符串的 q-gram 距离提出一种将数据映射为 q-gram 空间中的点,然后使用层次聚类定位潜在重复记录来解决重复检测问题.但此方法针对字符串的 q-gram 距离,有一定的局限性.庞雄文等人[7]根据概念依赖图计算表的关键属性并据此将数据划分为记录集进行重复记录检测,采取一种合并已匹配记录的方法来减少比较的次数.但他们的工作没有考虑到重复往往缺乏传递性,即记录 A,B 重复,记录 B,C 重复,并不意味着记录 A,C 重复,因此此方法在实际运用中效果往往不够理想.Fan 等人[8]研究了待重复检测数据集中记录属性,提出了与联系相关属性间的一种新依赖关系,并提出了一种寻找这些依赖及关联候选键(relative candidate keys)的算法.此方法研究重点在于如何找到关键属性,对如何在大数据环境下高效地进行重复检测未有涉及.当前普遍采用的方法为 Hernandez 和 Stolfo 提出的 SNM 方法及一系列基于 SNM 的方法[3,9-10].

1.2 SNM 方法及其改进

SNM 方法的核心思想是利用一个排序函数对整个数据集中的数据进行排序,将潜在可能重复的记录排在一起(分块).然后利用窗口技术对数据对比的空间进一步压缩.传统的 SNM 方法分为 3 个步骤:

1) 键值指派.在此阶段,为每一条记录指派一个用于排序的键值.通常,这个键值是根据属性值的某些部分生成的.

2) 排序.所有记录按键值进行排序.

3) 窗口化.一个固定长度的窗口在已经排序的数据上滑动.窗口内的每一对数据都两两进行比较,然后标识出重复数据.

现实世界中重复记录数目是随机的,可能会有很多条重复记录,也可能仅有 1,2 条重复,而使用一个固定大小的窗口限定比较范围,显然并不合适.Yan 等人[9]基于一个局部的距离单调假设提出了动态窗口的方法.Draisbach 等人[10]则基于重复记录密集的地方可能存在更多的重复记录这一假设提出在重复检测过程中根据窗口内检测到的重复记录数来动态调整窗口大小,以提高比较效率的 DCS 方法和 DCS++方法.与 SNM 相比,DCS 及 DCS++的效率有显著提高.

1.3 重复检测问题的不足以及本文的工作

仔细分析 SNM 类方法,可以发现如下不足:SNM 类算法基于键值指派,排序函数选择对结果影响明显,好的排序函数能有效降低比较的次数.然而在实际应用中,找到合适的排序函数并不是一件容易的工作;而且排序函数的实质是将高维数据映射到一维空间上,在映射的过程中不可避免地会导致部分信息的损失,从而使得排序结果与实际存在一定的偏差,数据的维度越高,信息损失越大,因此 SNM 类算法并不能很好地伸缩.此外,重复检测采取一种窗口内所有记录逐对比较的策略,在对比的过程中并没有或者仅仅是基于假设有限利用之前判断的结果,这样做大大降低了重复检测的效率.

为解决 SNM 类算法在处理高维数据时上述的不足,本文提出了一种基于聚类思想的高维数据重复检测算法,保留高维数据信息并建立索引用于重复检测,在检测过程中有效地利用了之前比较的结果降低比较次数.由图 1 的重复检测框架可知,对象之间是否重复是根据相似度向量来判断的,因此相似度的定义对最终算法的效率有较大的影响,为使讨论更加清晰,本文不失一般性采用了欧氏距离作为相似度判别标准,利用记录间距离是否达到一个阈值 t 来判断记录对是否重复.

2 基于空间邻居的重复检测

2.1 不确定性与重复检测

现实生活中应用能获得的数据与真实数据之间往往存在一定的差异,测量仪器的误差、人为疏忽而导致的输入错误、基于隐私保护考虑而引入的扰乱等原因都会造成真实数据与观察值之间的差异,这种差异被称为不确定性.数据的不确定性可分为2类:存在级层次的不确定性(existential level uncertainty)和属性层次的不确定性(attribute level uncertainty).其中后者是一个研究得较多的不确定模型,它假设对象某属性的实际数据分布于一个封闭的不确定区间中,数据的出现服从一个该区间内的概率密度函数.

本文认为,不同数据源之间会出现重复记录的现象可视作不确定性在现实世界中的体现.由于存在不确定性的影响,同一客观对象在每个数据源中观察到的属性值会存在一定差异,这就导致了重复记录的产生.因此,对数据集进行重复检测的目的可视为希望通过重复检测找到一个更接近现实世界的数据集.

2.2 重复检测与聚类

重复检测是在一个大型数据集中寻找类似记录的过程,这与数据挖掘中的聚类分析十分相似.聚类分析是一个将物理或抽象对象集分成相似对象类的过程,簇是相似数据对象的集合,聚类分析要求同一个簇中的对象彼此相似而与其他簇中的对象相异.如果将同一个客观对象的重复记录看成一个簇,那么重复检测问题可以看成为一个特殊的聚类问题.

但在一般意义上,重复检测问题并不仅仅是一个聚类问题,它是一个比一般聚类要求更严格的数据聚集问题.比较重复检测过程和传统的聚类算法可以发现:前者关注的重点是具体的记录对,属于同一个客观对象的重复记录两两之间的距离/差异必须小于一个特定的阈值;而后者关注的重点是簇.一般地,在寻找簇的过程中并不会特定关注簇中每一对对象是否相似,常用的聚类算法如 K-均值和BRICH等,并不保证簇内每一对对象是相似的,基于密度的聚类方法仅要求簇内的密度达到一个阈值就可以,若簇足够大,那么处于同一个簇内边界2端的对象之间的差别是非常大的.可见,传统的聚类方法并不能直接应用于解决重复检测问题.

然而,对比聚类分析及重复检测的结果可以发现:虽然使用聚类算法发现的同一簇内的对象不一定重复,但属于同一个客观对象的重复记录必定在一个簇中.因此可以利用聚类分析来有效提高重复检测的效率.

2.3 利用 R-树进行高维索引

R-树是B-树在多维空间的扩展,是目前应用最为广泛的一种空间索引结构,它按空间存储数据,能有效提高高维数据的聚类效率.R-树中有2种结点:叶子结点和中间结点、叶子结点记录数据;中间结点记录其子树的索引空间.当需要查找某一空间点的近邻时,可以利用中间结点将搜索范围限制在一个比较少的范围.一棵 M 阶的 R-树,其结点结构可以描述如下:

叶子结点:

$(Count, Level, \langle OI_1, MBR_1\rangle, \langle OI_2, MBR_2\rangle, \cdots, \langle OI_M, MBR_M\rangle)$,

中间结点:

$(Count, Level, \langle CP_1, MBR_1\rangle, \langle CP_2, MBR_2\rangle, \cdots, \langle CP_M, MBR_M\rangle)$,

其中,$Count$ 为结点的索引项或数据项的个数,$Level$ 表示结点类型,$\langle OI_i, MBR_i\rangle$ 称为数据项,OI_i 是第 i 个空间目标的标识,MBR_i 为该目标在 k 维空间中的最小包围矩形(简称为数据矩形);$\langle CP_i, MBR_i\rangle$ 称为索引项,CP_i 是指向子树根结点的指针,MBR_i 代表其子树索引空间,为包围其子树根结点中所有目录矩形或数据矩形的最小包围矩形(简称为目录矩形).

3 高维数据的快速重复检测算法

3.1 本文使用的重复检测处理框架

影响重复检测效率的主要原因在于索引的效率以及记录对的比较次数,为使讨论更加清晰,在本文讨论中简化了相似度向量分类步骤,同时本文的讨论中不对数据的具体类型、分布和依赖关系做约束,不失一般性,假设待处理数据集为 n 维向量集,利用2条记录间的欧氏距离是否达到一个预定义的阈值 t 来判断记录对是否重复.本文将重复检测处理框架简化如图2所示的流程.

在图2所示的重复检测过程中,对来自不同数据源的数据清洗和标准化处理之后,本文提出的DDR算法利用R-树作为空间索引,然后经过一次遍历将可能的记录对进行比较,最后输出结果.

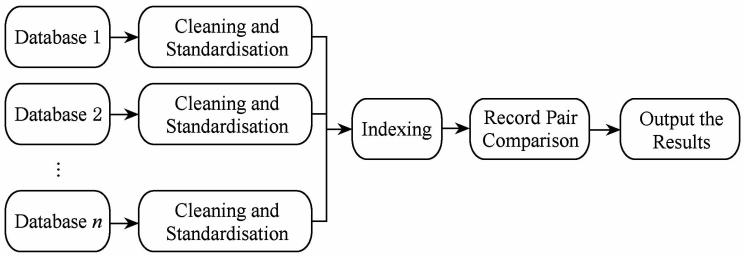


Fig. 2 Duplicate detection framework proposed.

图 2 本文算法所采用的重复检测框架

算法 1. DDR 算法.

输入:待处理数据集;

输出:重复记录集.

DDR(*Dataset*){

① *Build_R_tree*(*R*,*Dataset*); /* 以 *R* 为根结点,建立 R-树 */

② *Traversal*(*R*,*t*,0); /* 以 *t* 为阈值遍历 R-树,输出重复记录 */

③ }

3.2 DDR 中的遍历

重复检测需要对所有潜在可能重复的记录对进行比较,一个简单的思想是可以将之前搜索的结果作为后续搜索的启发. 在 DDR 的遍历中记录了数据查找及对比过程中的相关信息,每次搜索都利用这些信息在已经进行过重复检测的数据中进行,这样做能有效地减少搜索空间、提高搜索效率.

为适应这种搜索策略,对 R-树进行了如下的修改:每层结点增加 2 个属性 *Visited* 和 *Contain*. *Visited* 代表已经访问过该结点. 该属性在 R-树初始化时记录为 0,若该结点为叶子结点,那么在叶子结点的所有数据都被处理后该属性记为 1,若该结点为中间结点,那么当其被遍历时记为 1. 属性 *Contain* 表示结点中是否可能包含与当前正在进行重复检测的叶子结点中的数据重复的记录. 当开始访问一个新的叶子结点时,先利用该结点的目录矩形对 R-树已经遍历过的部分进行一次遍历,在遍历过程中利用距离阈值对各个结点的属性 *Contain* 进行标注,如果两者的目录矩形相交或距离小于阈值 *t*,那么意味着该结点中可能包含重复记录.

DDR 中使用一个递归的遍历算法,其基本思想如下:如果访问到一个叶子结点,那么依次将该结点中的数据项与之前已经进行过重复检测的数据项进行对比,在对比的过程中利用 R-树的结构可有效地压缩搜索范围,然后将该数据项与叶子结点内部的数据项对比,在对比的过程中不断优化叶子结点数据项的排列.

算法 2. DDR 中的遍历算法.

Traversal(*R*,*t*,*MBR*){

① if *R*.*Level*=0 then {/* 若 *R* 是叶子结点遍历一次 R-树,利用目录矩形标识已访问过的结点中可能包含重复记录的结点 */

② *Check_Visited*(*Root*,*MBR*,*t*);

③ for *i*=1 to *R*.*Count* {/* 对叶子结点中每个数据项 *D*,搜索所有已访问过且与 *D* 潜在距离 *t* 相交的叶子并保存在集合 *data* 中 */

④ *D*=*R*.*MBR*$_i$;

⑤ *data*=*R_Search*(*Root*,*D*,*t*);

⑥ Compare *D* with data in every leaf in *data* and output the duplication records;

⑦ *Compare_inside_leaf*(*R*,*i*,*t*); /* 在叶子结点内部进行逐一比较,并为叶子结点内部数据建立一个便于查找的序 */

⑧ } end for

⑨ *R*.*Visted*=1;

⑩ } else {/* 如果不是叶子结点,那么依次遍历他的各个子树 */

⑪ *R*. *Visted*=1;

⑫ for *i*=1 to *R*.*Count*{*traversal*(*R*.*CP*$_i$,*t*,*R*.*MBR*$_i$);

⑬ } end for

⑭ } end if }

3.3 在叶子结点内部进行比较

R-树中的数据项皆保存于叶子结点内,由于 R-树结构的特点,每个叶子结点内的数据项之间位置相近是潜在重复的. 因此,有必要对叶子结点内的数据项进行两两比较. 与 R-树搜索策略类似,DDR 在叶子结点内部比较过程中记录部分信息以提高后续的对比效率.

由 2.3 节中讨论可知,本文将重复检测看成一个特殊的聚类问题,相同实体对象的重复记录将聚集成一个簇,簇中记录两两间的距离小于阈值 t,而簇外记录至少与簇中一个记录的距离大于 t. 以图 3 为例,重复的判定阈值为 t,点 a 属于一个簇 A,显然 a 与簇内任一点皆重复;点 b,c 与 a 的距离大于 $2t$,显然,b,c 与 A 中任一点皆不可能是重复的;而点 e 与 a 的距离大于 t 小于 $2t$,它与簇 A 中其他点可能重复也可能不重复. 直观地可得到如下的性质:

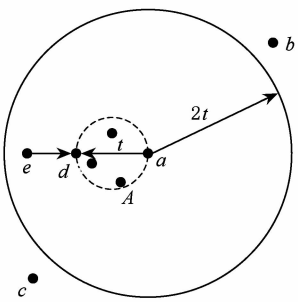


Fig. 3 Relation between points in the cluster and out of the cluster.

图 3 簇外点与簇内点的关系

性质 1. 如果一个记录 D 与一个簇 C 中任一记录距离大于 $2t$,那么 D 与簇中任何记录都不可能组成重复记录对.

根据此性质,DDR 算法可以尽快判断哪些点肯定不在同一簇中,从而避免了相关的对比. 该算法对叶子结点内的重复检测过程如下:为 R-树叶子结点中每个数据项增加一个属性 S,它代表了滑动的步长. 如果一个数据项的 $S=n$,即意味着它后续的 n 个数据项与该数据项处于同一簇中,当叶子结点内部进行比较时,若记录 D 与一个数据项 OP 的距离大于 $2t$,那么意味着 D 与 OP 及其之后 n 个数据项都不可能是重复记录. 具体算法如算法 3 所示.

算法 3. DDR 中叶子结点内部比较算法.

```
Compare_inside_leaf(R,i,t) {
① D=R.MBR_i;
② Is_belong_to_a_cluster=0;
③   for j=1 to i-1 {
④   dist=distance(D,R.MBR_j);
⑤   Choose case dist /* 根据数据项之间的距离分别做判断 */
⑥   case dist≤t /* 在阈值 t 之内的,输出重复记录对,同时在簇内进行扫描,判断数据项是否属于该簇并进行对应处理 */
⑦     Output(D,R.MBR_j);
⑧     Is_in_this_cluster=1;
⑨   for k=1 to R.S_j{/* 对簇内每对数据项判断 */
⑩     if distance(D,R.MBR_j+k)>t then
        {Is_in_this_cluster =0;}
      else {Output(D,R.MBR_j+k);} end if
⑪   } end for
⑫     if Not(Is_belong_to_a_cluster) && (Is_in_this_cluster) then {
⑬       把簇之后的数据项往后移一位;
⑭       将该簇内所有数据项的步长加 1;
⑮       将此数据项移至簇的末尾;
⑯       Is_belong_to_a_cluster=1;
⑰     } end if
⑱     j+=(R.S_j+1);
⑲   case dist>2t /* 大于 2 倍阈值,直接根据步长滑动 */
⑳     j+=R.S_j;
㉑   case else /* 在 1 倍阈值与 2 倍阈值之间的,按平常做法依次扫描簇内的数据项 */
㉒     for k=1 to R.S_j{
㉓       if distance(D,R.MBR_j+k)≤
           t{Output(D,R.MBR_j);} end if
㉔     } end for
㉕     j+=R.S_j;
㉖   end case
㉗ } end for }
```

下面以图 4 所示叶子结点为例,讨论 DDR 算法叶子结点内重复检测的过程. 该叶子结点的初始状态如图 4 中 Step1 所示,结点所包含的前 6 个项的值分别是 5,3,10,11,4,2. 给定阈值 $t=2$,在开始检测之前,所有项的步长记为 0.

Step1. 从项 2 开始,值 3 与值 5 间距离为 t,它们属于一个簇,输出重复记录对(3,5),同时项 1 的步长改为 1,意味着它与其后的项在一个簇中.

Step2. 从项 3 开始,值 10 与值 5 距离大于 $2t$,而项 1 步长为 1,因此无需与项 2 进行比较.

Step3. 从项 4 开始,值 11 与值 5 距离大于 $2t$,项 1 步长为 1,因此无需与项 2 比较,项 4 直接与项 3 比较,值 11 与值 10 距离小于 t,它们属于一个簇,输出重复记录(11,10),同时项 3 步长改为 1.

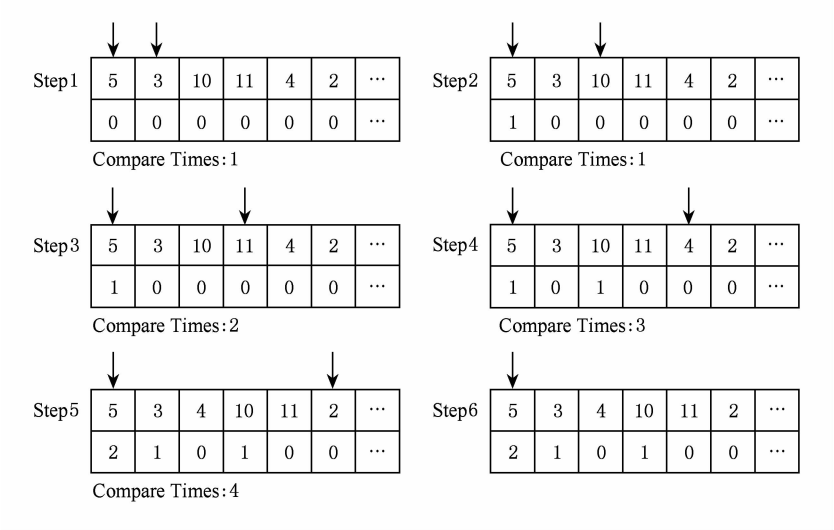


Fig. 4 An example of traveling in the leaf node of R-tree.

图 4 一个 R-树叶子结点遍历的例子

Step4. 从项 5 开始,值 4 与值 5 距离小于 t,输出重复记录对(4,5),因为项 1 步长为 1,所以需要继续与项 2 比较. 值 4 与值 3 距离小于 t,输出重复记录对(4,3),可以发现此 3 项两两互为重复记录,因此它们形成一个簇,算法进行如下操作:将项 5 的位置移至第 3,原来的项 3、项 4 后移一位变成项 4、项 5,同时将项 1,项 2 步长各加 1. 然后继续比较. 项 3 的值 4 与项 4 的值 10 之间距离大于 $2t$,项 4 步长为 1,因此项 3 与项 5 无需比较.

Step5. 从项 6 开始,值 2 与值 5 距离大于 t 小于 $2t$,因此不能利用步长减少比较次数,项 6 与后续项 2、项 3 进行比较,发现都是重复项,输出(2,3),(2,4),但此 2 项皆与项 1 处于同一簇内,因此不需要对其进行进一步处理. 继续项 6 与项 4 的比较,值 2 与值 10 距离大于 $2t$,因此不必与项 5 比较.

3.4 DDR 算法分析

从 3.1 节讨论中可知,DDR 算法可以分为相对独立的 2 部分:构建 R-树以及利用 R-树查找重复记录. 下面分别对其效率进行讨论:

传统的 R-树的构建方法称为 OBO(one-by-one)方法,即从空树开始,利用插入算法逐个插入记录,直至生成整个 R-树. 由于插入过程中需要保持 R-树的动态平衡,所以建树代价较高. 当需要建立索引的数据已知且相对静态时,可以采取批量加载的方法得到结构优化的 R-树,这样能更快建立 R-树并保证其在查询时能有更好的表现. 一般地,重复检测问题是对已知数据进行批量检测,因此,本文实验中选取了一个时间复杂度为 $O(Sn\log n)$ 的批量加载方法 TGS 建立 R-树[11],其中 S 是一个与处理数据集相关的参数.

利用 R-树查找重复记录的时间消耗主要在对记录进行对比时的距离计算上,其比较次数与预定义的阈值有关,最坏情况下需要做 $n(n-1)/2$ 次比较. 与建树阶段相比,查找重复记录阶段不仅有更高的时间复杂度,而且每一次比较也需要更多的单位时间. 因此,整个算法的时间消耗主要与记录对比次数相关,这与 SNM,DCS,DCS++ 等算法是一致的. 本文的第 4 节将通过实验数据详细比较 DDR 建树和搜索 2 阶段的效率以及 DDR 与 SNM,DCS,DCS++ 等算法的效率.

3.5 利用 Apriori 性并行地高速处理高维数据

利用 R-树索引的 DDR 算法能有效处理一定维度范围内的高维数据,然而高维数据本质是稀疏的,当维度数目足够大时,必然会面对维灾难的问题,R-树基于空间索引的优势将不再有效,其索引代价也不再可忽略. 为解决这种超高维度数据的重复检测问题,我们对此问题进行了深入的分析,发现高维数据在重复检测中具有如下性质.

性质 2. 如果一对数据在高维上是重复的,那么它们在低维空间上的投影一定也是重复的(apriori 性).

显然,若记录在各维度投影的数值不变,记录间距离随维度增加单调递增. 因此,若以 t 为阈值进行重复检测,由此性质可知:若一对高维记录在某一低维空间上的投影不重复(距离大于阈值 t),那么它们在高维空间中也不可能是重复的(距离小于等于阈值 t). 为此,本文提出了一种并行地利用数据集

在低维空间中的投影对其进行过滤,生成候选集来缩减搜索空间的方法.具体思想如下:对数据集在若干低维组合的投影进行重复检测,利用 Apriori 性剔除大部分不可能重复的记录对,得到潜在可能的重复记录集,然后在高维下对这些潜在重复的记录对进行检测.

由于 Apriori 性质非常适合于并行处理,因此本文采用了一个 Map-Reduce 的方法来高效地实现此过程,其中 Map-Reduce 函数如下:

Map 函数.每一个具体的 Map 函数实质上就是一个利用阈值 t 对低维子空间中的数据投影集进行重复检测 DDR 算法,其输出为每个数据块上潜在可能的重复记录对.Map 函数将原数据集在低维投影上距离小于阈值 t 的记录对组合编码为键,值记为 1,并输出键-值对作为 Reduce 函数的输入.

Reduce 函数.对 Map 函数输出的结果中具有相同键值的键-值对进行计数,将计数值等于 n(Map 工作机数目)的键值输出,它所对应的记录对就是潜在重复的记录对.

3.6 大数据环境下高维数据的重复检测

利用 Apriori 性质的 Map-Reduce 方法能有效减少高维数据对计算的影响,但值得注意的是,在每一台 Map 工作机上运行的是一个独立的 DDR 算法.从 3.4 节分析中可以看到,最坏情况下 DDR 要进行 $n(n-1)/2$ 次比较,在大数据环境下这显然是不可接受的.

深入分析 SNM 类算法在重复检测中的作用可以发现,虽然使用散列排序函数会造成大量信息损失,不能为后续计算提供便利,但散列排序函数将大数据集分割为若干个小数据集缩减比较次数的方法能有效地减少数据集规模增长所带来的计算量.为此,本文采用了如图 5 所示过程对大数据环境下的高维数据进行重复检测.

首先根据具体的距离定义利用散列函数将数据分割为若干相容的子集.每个子集分为 2 部分,核心区域以及扩展区域.全体核心区域的并集等于整个数据集,而且与核心区域中数据的潜在重复记录都落在核心区域及其对应的扩展区域中.图 6 所示 1 维数据集重复记录距离小于阈值 l,实线方框所示为核心区域,虚线所表示为对应扩展区域,显然核心区域记录所对应的重复记录必包含在其扩展后的区间内.

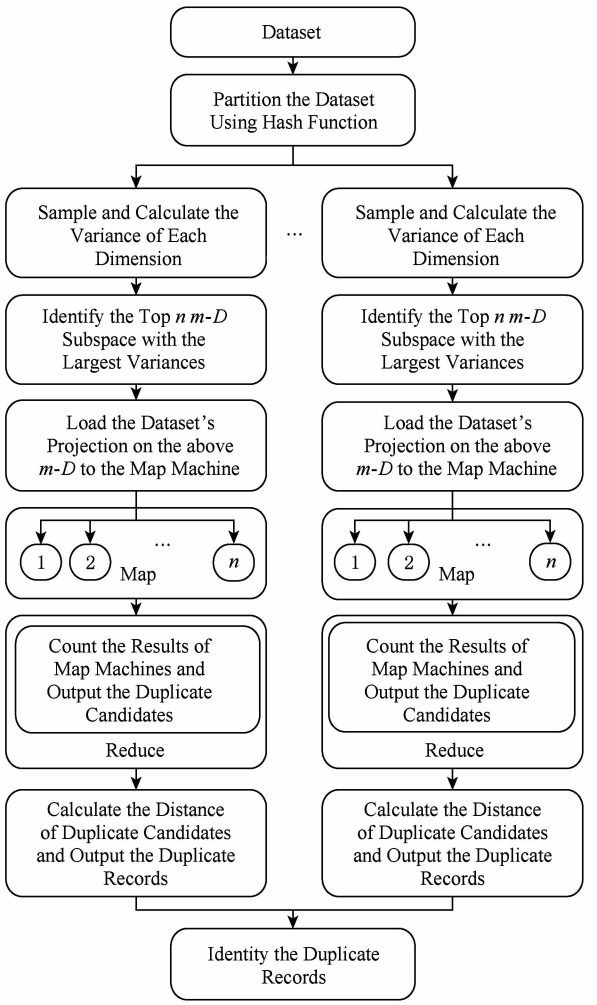


Fig. 5 Apriori property based duplication detection for high dimensional data.

图 5 利用高维数据的 Apriori 性质进行重复检测

Fig. 6 Kernel area and expansion area.

图 6 核心区域与扩展区域

虽然没有一个简单的方法预测数据在哪些子空间上具有较低的重复度,但是一般而言,数据在某维上投影的方差越大,意味着数据在该维上的分布越分散.因此,可以利用抽样技术计算其在各维上的方差,然后将数据集分成 n 块(n 为 Map 工作机的数目),每一块中记录了整个数据集在方差最大的 n 个 m 维空间上的投影(m 为 DDR 算法最佳的收敛空间),利用数据在这些维度上的投影和一个 Map-Reduce 过程对数据集进行初步处理,得到潜在的重复记录对.最后根据定义对这些潜在重复记录对进行判断并输出最终结果.

4 实验及结果分析

4.1 实验方法与实验数据集

本文的实验分为2部分:1)比较了DDR算法的有效性及其与SNM,DSC,DSC++算法在处理高维数据时效率的差别;2)通过实验检验了本文提出的Map-Reduce过程对高维数据处理的效率和有效性.

本文所有实验使用的代码均为Java实现,同时为了使比较的数据集更具有可信性,本文选用了参考文献[10]中所使用的Febrl数据集的数据生成器[12]人工生成了本文实验中所使用的数据.

4.2 DDR算法实验结果及分析

Febrl数据集中每条数据包含的信息为人的姓名、年龄、国籍、住址、电话…等18项信息.为更全面衡量算法的效率和伸缩性,本文对Febrl数据集中的数据进行标准化后,分别抽取其中2维、4维、8维和10维信息生成了100万条数据和1万条数据2种不同大小的数据集进行重复检测.

每个数据集中的数据由2部分组成,其中50%为原始数据,另外50%为根据原始数据做改动的重复记录.重复记录的生成规则如下:对原始记录每一维的数据进行标准化处理,每条原始数据有0~9条对应的重复数据,重复数量服从Zipf分布;每条重复数据有0~5处改动.

首先比较DDR算法中构造R-树阶段以及利用R-树进行搜索阶段时间的消耗.图7显示了在不同维度的数据集中使用DDR算法进行重复检测时建树阶段和搜索达到一定召回率所需的时间.在此实验中,我们对4个不同维度各100万条数据分别进行建树和搜索.从图7可以看出,要达到95%的召回率,针对4维、8维和10维3个数据集进行重复检测所需时间与建树时间接近,但它们建树和搜索时间之和远小于2维数据集搜索时间所需,这是因为利用R-树能有效地减少候选集,其带来的效益提高远大于建树所消耗的时间.从图7还可以看出,要达到100%的召回率,4种不同维度数据所需搜索时间均远超其建树时间.事实上,在我们的实验中,当召回率足够大时,DDR算法中构造R-树阶段所需时间消耗基本可以忽略.

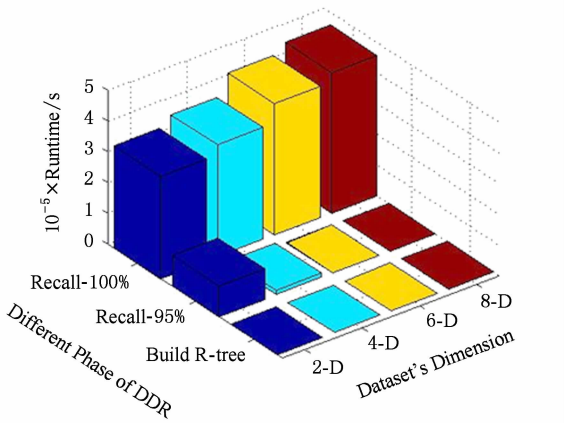


Fig. 7 Comparison of time for different DDR phases.

图7 DDR算法各阶段效率比较

重复检测算法的时间消耗主要用于对比记录,因此本文通过不断调整匹配阈值的方法来分别检测4种方法达到一定的召回率所需要的比较次数.考虑到SNM,DCS,DCS++算法在处理大批量数据时表现不佳,在本次实验中分别对2维、4维、8维和10维的数据集使用了1万条记录进行测试.表2显示了DDR算法与SNM,DCS,DCS++算法比较的部分结果,图8则显示了4种算法在不同维度、不同召回率之下所需对比次数的曲线,从实验结果可以看出,虽然4种算法在不同维度的数据集中的表现有所差别,但一般地,达到同样的召回率DDR所需的比较次数少于SNM,DCS,DCS++算法.

Table 2 Partial Results of Number of Comparison on 10-D Dataset

表2 10维数据集中部分比较次数

Recall/%	SNM	DCS	DCS++	DDR
90	20 330 747	18 348 090	11 357 313	60 428
95	35 968 544	35 431 209	23 957 136	185 818
98	44 024 760	44 920 807	31 531 092	711 273
99	44 940 390	46 783 683	33 270 645	2 372 714
100	49 957 325	49 974 947	35 955 043	32 379 635

观察实验结果可以发现,当召回率低于98%时,DDR方法优势明显,而当召回率达到99.9%时,DDR算法的优势大大削弱,这是因为本文所采用的实验数据中的重复记录是随机生成的,某些记录会与它的重复记录有较大差距,当召回率接近100%时,所需要搜索空间会比较大,接近于全局搜索.另外,数据集的维度越高,DDR的优势越突出.DDR在2维数据集中的表现与DSC++相比优势并没有其在高维数据集中明显,这是由R-树对候选集的裁剪在高维时更明显所导致的.

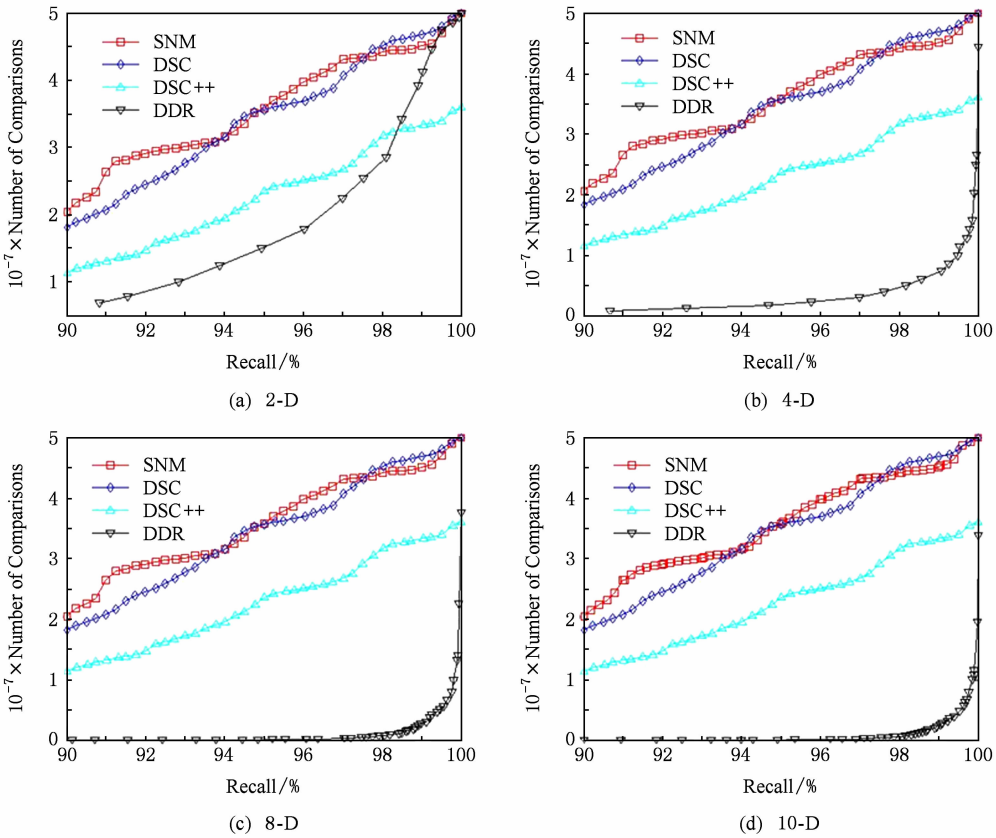


Fig. 8 Efficiency of four algorithms on datasets of 2-D,4-D,8-D,10-D.

图 8 4 种算法在不同维度的数据集中的效率比较

为了更有效地检验算法的伸缩性,本文针对 2 维、4 维、8 维和 10 维的数据集使用了 100 万条记录对 DDR 进行测试,比较结果如图 9 所示. 从图 9 可以看出,随着维度的增加,DDR 所需比较次数明显减少,但当增加到 8 维以上,虽然比较次数依然减少但已不再明显,这是由高维数据的稀疏性决定的. 另外,虽然不同维度的数据集在比较次数有明显的差别,但所消耗时间的差异并没有如此显著,这是因为相对于高维数据,低维数据在计算距离时所消耗的时间相对较少所导致的.

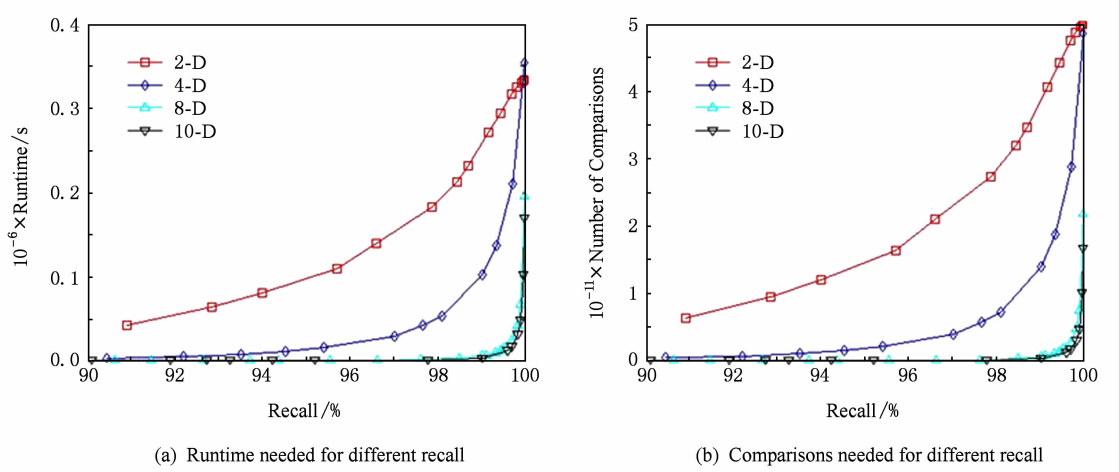


Fig. 9 Efficiency of DDR on datasets of 2-D,4-D,8-D,10-D.

图 9 DDR 在不同维度的数据集中的效率比较

4.3 高维数据处理的效率

本文提出的大数据环境下重复检验框架效率取决于 Apriori 性的剪枝效率,而此效率与重复记录的距离阈值紧密相关. 为检验剪枝效率,本文设计实验

如下:使用 4.1 节中生成重复记录的方法生成一个 10 万条 100 维的数据集,利用一个有 10 个 Map 工作机的 Map-Reduce 系统对其进行重复检测,根据上文讨论结果,每台 Map 工作机中仅处理数据在其某 10 维上的投影.实验通过调整不同的重复记录距离阈值,衡量剪枝效率,结果如表 3 所示:

Table 3 Experimental Results of Pruning on 100-D Dataset

表 3 剪枝效率实验结果(100 维数据集)

Distance	Average Number of Record Pairs in Each Map Machine	Number of Duplicate Record Candidates	Number of Duplicate Records
5	935 344.3	67 167	9 743
6	4 644 185.5	374 204	12 175
7	17 229 624.4	1 670 758	16 606
8	43 784 665.3	5 375 299	22 037
9	85 888 457.7	12 558 183	29 037
10	146 417 119.1	25 913 531	37 520

从表 3 可以看出,当距离阈值在一定范围时,利用 Apriori 性质剪枝后得到的潜在重复记录对数目远小于平均每个 Map 工作站得到的低维重复记录数.因此,本文提出的剪枝方法能有效减少需要进行重复检测的潜在记录对数目.

图 10 显示了随距离阈值增加剪枝效率改变的曲线.从图 10 可以看出,随着距离阈值增加,全体 Map 工作站结果能组成的潜在重复记录对在每个 Map 工作站所得结果中占比越来越高,但实际重复记录对在潜在重复集中所占比例不断降低.这意味着,随着重复判定的距离阈值增加,利用 Apriori 性质剪枝效果逐渐衰减,这是由高维空间数据的稀疏性所导致的.

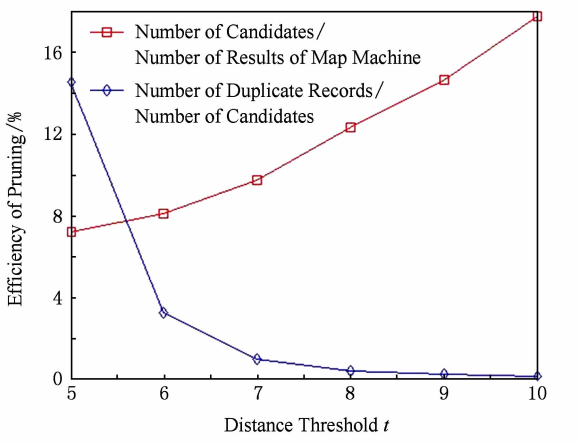


Fig. 10 Pruning efficiency using Apriori property.

图 10 利用 Apriori 性质剪枝效率

5 结束语

从第 4 节的实验结果中可以看出,DDR 算法在处理高维数据时明显优于传统的重复检测算法,而且当数据维度小于 10 维时,维数越高算法优势越明显.然而,从实验中也可以看出 DDR 算法仍有如下不足:当面对高维数据时,若要求达到 99%以上的召回率,DDR 的时间消耗仍然很大.这是因为重复检测需要计算每一对潜在重复记录之间的距离,所以即使 DDR 算法大大裁剪了候选重复记录集的数目,仍需进行大量的计算.因此,如何针对不同的数据集确定一个合适的距离从而在召回率和比较次数之间达到一个合理的平衡有待进一步研究.我们正在研究如何利用抽样及软件测试中的缺陷率估计技术等方法为真实数据集确定一个合适距离.

从剪枝实验结果可以看出,当数据维度较高时,利用 Apriori 性质和一个 Map-Reduce 过程进行剪枝能有效减少需要计算距离的潜在记录对.但随距离阈值增大剪枝效率会降低.因此,如何为高维数据挑选一个合理的距离阈值,从而在召回率和计算效率之间达到一个有效的平衡,也将是我们进一步研究的方向.

参 考 文 献

[1] Meng Xiaofeng, Ci Xiang. Big data management: Concepts, techniques and challenge [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169 (in Chinese)
(孟小峰, 慈祥. 大数据管理: 概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-169)

[2] Li Jianzhong, Liu Xianmin. An important aspect of big data: Data usability [J]. Journal of Computer Research and Development, 2013, 50(6): 1147-1162 (in Chinese)
(李建中, 刘显敏. 大数据的一个重要方面: 数据可用性[J]. 计算机研究与发展, 2013, 50(6): 1147-1162)

[3] Ahmed K E, Panagiotis G I, Vassilios S V. Duplicate record detection: A survey [J]. IEEE Trans on Knowledge and Data Engineering, 2007, 19(1): 1-16

[4] Christen P. A Survey of Indexing techniques for scalable record linkage and deduplication [J]. IEEE Trans on Knowledge and Data Engineering, 2011, 24(9): 1537-1555

[5] Hernandez M A, Stolfo S J. The merge/purge problem for Large database [C] //Proc of the 1995 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 1995: 127-138

[6] Han Jingyu, Xu Lizhen, Dong Yisheng. An approach for detecting similar duplicate records of massive data [J]. Journal of Computer Research and Development, 2005, 42(12): 2206-2212 (in Chinese)
(韩京宇, 徐立臻, 董逸生. 一种大数据量的相似记录检测方法[J]. 计算机研究与发展, 2005, 42(12): 2206-2212)

[7] Pang Xiongwen, Yao Zhanlin, Li Yongjun. Efficient duplicate records detection method for massive data [J]. Journal of Huazhong University of Science and Technology: Natural Science Edition, 2012, 38(2): 8-11 (in Chinese)
(庞雄文, 姚占林, 李拥军. 大数据量的高效重复记录检测方法[J]. 华中科技大学学报: 自然科学版, 2012, 38(2): 8-11)

[8] Fan W, Gao H, Jia X, et al. Dynamic constraints for record matching [J]. The VLDB Journal, 2011, 20(4): 495-520

[9] Yan S, Lee D, Kan M, et al. Adaptive sorted neighborhood methods for efficient record linkage [C] //Proc of the 7th ACM/IEEE-CS Joint Conf on Digital Libraries (JCDL'07). New York: ACM, 2007: 185-194

[10] Draisbach U, Naumann F, Szott S, et al. Adaptive windows for duplicate detection [C] //Proc of the 28th Int Conf on Data Engineering (ICDE'12). Piscataway, NJ: IEEE, 2012: 1073-1083

[11] Alborzi H, Samet H. Execution time analysis of a top-down R-tree construction algorithm [J]. Information Processing Letters, 2007, 101(1): 6-12

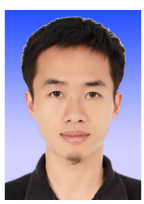
[12] ANU Data Mining Group. Prototype software [CP/OL]. 2014 [2014-06-01]. http://datamining.anu.edu.au/linkage.html

Zhu Weiheng, born in 1976. Lecturer. Receive his PhD from Sun Yat-sen University. His main research interests include database, data mining and pattern recognition.

Yin Jian, born in 1968. Professor and PhD supervisor at Sun Yat-sen University, China. Senior member of China Computer Federation. His main research interests include big data and data mining.

Deng Yuhui, born in 1974. PhD, professor and PhD supervisior. His current research interests cover data storage and data management, cluster computing, and green computing.

Long Shun, born in 1971. Associate professor at Jinan University, China. His main research interests include machine learning, data mining and information retrieval.



Qiu Shiding, born in 1989. Received his master degree at Jinan University in 2014, China. His main research interests include social network mining, large scale machine learning.