基于分块和滑窗技术的相似重复记录检测算法研究

陈 亮 杜 璐* 胡 康
(西安工程大学计算机科学学院 陕西 西安 710048)

摘 要 相似重复记录检测对于提高数据质量有着重要意义。为了减少检测代价和提高运行效率,基于传统的窗口技术和分块技术,提出一种相似重复记录检测算法。该算法利用关键字段将数据集进行排序和分块,并利用滑动窗口技术限制分块间比对。设计一种多字段排序改进算法,对不同字段的分块共同聚类,优先比较重复密度大的分块对,摒弃聚类较差的分块。该算法减少了检测过程中的数据比较次数,并降低了字段好坏对算法速度的影响。理论和实验分析表明,该算法能有效地提高相似重复记录检测的准确率和时间效率。

关键词 数据质量 相似重复记录检测 窗口技术 分块技术

中图分类号 TP311 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.04.042

AN DUPLICATE DETECTION APPROACH BASED ON BLOCKING AND WINDOWING

Chen Liang Du Lu* Hu Kang
(*School of Computer Science, Xi'an Polytechnic University, Xi'an 710048, Shaanxi, China*)

Abstract Duplicate detection plays an important role in data quality. In order to reduce the detection cost and improve the algorithm efficiency, we proposed an effective duplicate detection algorithm, which was based on the traditional windowing and blocking. The algorithm sorted data sets by keyword and partitioned data into multiple blocks. And windowing technology was applied to restrict comparisons between blocks. We designed an improved multi-key sorting algorithm. It clustered different key together, gave priority to the block pairs with high repetition density and discarded the blocks with poor clustering. The improved algorithm reduced the number of data comparisons in the detection process, and reduced the impact of field quality on the speed of the algorithm. Theoretical and experimental analyses shows that it can effectively improve the accuracy and time efficiency of duplicate detection.

Keywords Data quality Duplicate detection Windowing Blocking

0 引 言

在大数据时代,数据成为有价值的公司资产,可以减少和消除企业经济活动中的风险,为企业的管理控制和科学决策提供合理依据,并预期给企业带来经济利益[1]。公司或企业为了更好地做出决策,往往需要高质量的数据[2]。但是由于各种原因,例如:重复输入、不同数据源中数据采用不同的表示方式、多数据源合并造成重复等数据质量问题,使数据中心中存在着很多的相似重复数据。这些"脏数据"导致了错误的分析结果,进而影响决策。因此数据清洗,尤其是数据清洗中的相似重复数据检测变得尤为重要。

相似重复记录检测就是识别一对记录是否表示为真实世界中的同一个实体。为了避免高代价的记录比对,减少无意义的数据比对次数,有学者提出了一种分区算法,将大量的数据分成更小的子集[3-11]。如果相似记录分布在同个子分区,在数据比对过程中仅与区

收稿日期:2018-10-15。陕西省工业攻关资助项目(2014K05-43);陕西省教育厅专项科研项目(14JK1310);广东省计算机集成制造重点实验室(CIMSOF2016001)。**陈亮**,副教授,主研领域:云计算与大数据处理,并行计算,计算机网络。**杜璐**,硕士生。**胡康**,工程师。

中的记录比对而不是全部数据[3]。常用的技术是分块技术[4]和窗口技术[5]。其中,分块技术将数据集分成互不相关的子集,在较小的子集内数据聚类;窗口技术通过滑动窗口选中固定大小的数据依次与目标数据来进行记录比对。本文结合了分块技术和窗口技术,提出了一种相似重复检测算法,减少数据比较次数,提高算法运行效率。

1　相关工作

在相似重复记录的检测方面已经有了一些成果。传统的"排序 & 合并"算法先将数据库中记录排序,然后通过比较邻近记录是否相等来检测完全重复记录。在传统的"排序 & 合并"思想的基础上,有人提出了近邻排序算法 SNM(Sorted Neighborhood Method)[5-15]、分块算法 BLOCKING[4] 和排序分块算法 SBM(Sorted Blocks Method)[3]。

SNM 算法根据选定的若干个属性字符串合并作为键对数据集排序,然后采用固定大小的滑动窗口进行聚类来识别相似重复记录。滑动窗口大小的选取难以控制,设定大了,势必增加比较的次数和时间,选取过小,又可能造成算法结果遗漏[6]。BLOCKING 算法采用"化整为零"的分区思想对数据排序、分块和聚类,分块间不做相似重复检测。分块的划分直接影响到算法的结果,如果相似记录都划分在同分区,算法有着较高运行效率和满意的结果,反之,算法大部分时间都是无意义的数据比对。SBM 算法结合了分块技术和窗口技术,利用滑动窗口选取数据分块并聚类。该算法能够通过滑动窗口技术改善了分块划分问题,但是仍存在一些问题:利用滑动窗口分块导致每个分块之间均有一定量的重复数据,导致了相同数据对多次比对的现象。

针对上述算法的问题,本文参考了 MPN 算法中滑动窗口思想和 BLOCKING 算法中的分块思想[12-17],提出了多排序字段分块近邻匹配算法(MBNM),目的是为了改善 BNM 算法中数据多次重复匹配现象,提高算法运行效率。

2　分块近邻匹配重复检测算法

由于多排序字段分块近邻匹配算法(MBNM)是基于分块近邻匹配重复检测算法(BNM),所以简单介绍 BNM 算法是非常必要。分块近邻匹配重复检测算法基础思想是:将较大的数据集分为较小的分块,以分块为基础单位完成分块之间的相互比较。优先比较相似数据较多的分块,而从达到在较短时间得到较完整的检测结果的目标。BNM 算法的计算过程如下:

(1) 选取合适的关键属性 K 初始化数据集 D 按升序排序得到新数据集,记作 $D_{\downarrow}^{K}$。

(2) 将排好序的数据集分成若干个大小相同的分块 b_n,用 r_i 来记录数据,分块尺寸为 S。

(3) 分块间比较,存在相似重复记录记为 1,否则记为 0。返回分块间对比的相似重复记录数。

(4) 比较分块延展值,根据设定的阈值判断是否进行二次延展。否则,停止比较,返回相似重复记录数。

(5) 合并分块内相似重复记录结果,从而得到最终的检测结果。

分块近邻匹配算法是基于"排序 & 合并"思想,算法只要分为排序阶段与检测阶段。算法比较分块对 (b_n, b_m) 检测方式有两种:一种是 $n = m$,比较次数为 $\frac{S^2 - S}{2}$;另一种是 $n \neq m$,比较次数为 S^2。总共比较次数:

$$C_{\text{total}} = \frac{n(s-1)}{2} + (w-1)\left(\frac{n}{s} - \frac{w}{2}\right)s^2 \qquad (1)$$

当分块大小 s 为 1 时,算法就是 SNM 算法,当窗口大小为 1 时,算法比较过程就是 BLOCKING 算法。因为 w、s 都是固定值,所以算法检测过程时间复杂度为 $O\left(\frac{n(2ws - s - 1)}{2}\right)$。

在实际情况中,最好的排序字段总是不知道或者是难以找到,这样导致 BNM 算法检验相似重复记录的结果正确性不高。

3　多字段分块近邻匹配算法

3.1　算法介绍

"多趟(multi-pass)"执行方法很好地解决了排序字段难找问题,就是多次执行单字段重复检测算法,每次采用不同排序字段排序。算法如果选取了"不好"的字段,在使用该字段的迭代过程,不但得不到较好的结果,还增加了许多无意义的数据比较次数,浪费了大量的运行时间[6]。

图 1 展示了 MBNM 算法的实现过程。比如说第 4 行第 5 列的方块表示第 4 分块中的全部记录和第 5 个分块中的全部记录的比对,方块中的"9"数字代表存在 9 对相似重复记录对。分块对(2,2)和(5,5)的相似重复记录数最大,表示着该分块与邻近分块数据更有可能是相似重复记录。接着算法对(2,2)和(5,5)

这两个较大的相似重复记录分块对延展。如图所示，算法选择上近邻块对和右近邻块对作为该分块对延展对，并比较新的分块对得到各分块对的相似重复记录数。重复上述过程直到所有的分块对比较结束。

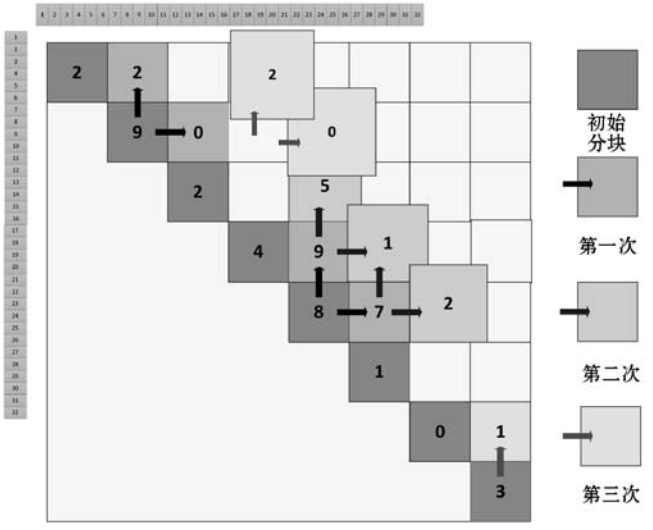

图 1 MBNM 算法演示图

3.2 算法描述

参考“多趟”执行方法中多排序字段的思想，提出多字段分块近邻匹配算法 MBNM(Multi-keys Blocking Neighbor Mapping)。改进算法也采用多个关键字段排序，但不是采用多趟执行的方式，而是多个排序字段一起执行。接下来结合伪代码详细介绍多字段分块近邻匹配算法(MBNM)的四个阶段：

(1) 数据预处理：

算法 1 数据预处理 dataPre(set)算法

输入：原始数据集 set

输出：处理后的数据集 D

算法对属性进行预处理。它将空格和标点符号视为分隔符，将属性的字符串拆分为单词，并按词法对单词进行排序。

(2) 多关键字定义与排序：

算法 2 多关键字排序算法

sortData(D,Ks,N,w,LowThreshold)算法

输入：待处理的数据集 D 和排序字段集 Ks

输出：多关键字排序后得到的新数据集 $D_{\downarrow}^{K}$

```
(1) bNum = ⌊N/w⌋
(2) flag = true
(3) for(int i = 1;i < bNum;i ++)
(4)     for(int j = 1;j <= w - 1;j ++)
(5)         for(v = j + 1;j <= w;j ++)
(6)             flag = match(Dj,Dv, LowThreshold)
(7)             if(flag == false)
(8)                 return fales;
(9) else return true
```

选取合适的关键属性 K 将待处理的数据集 D 按升序排序得到新数据集，用 $D_{\downarrow}^{K}$ 表示。接着将已排序的数据集分成若干的分块 b_n，每个分块都包含数量相等的数据记录(最后一个分块较小)，数据记录用 r_i 表示，分块尺寸为 w。

$$D_{\downarrow}^{K} = \left\{ b_n \,\middle|\, n = 1,2,3,\cdots,\left\lfloor \frac{N}{w} \right\rfloor \right\} \tag{2}$$

$$b_n = \{ r_i \mid i = 1,2,3,\cdots,w \} \tag{3}$$

在每次循环迭代中，算法首先会选取$\frac{bNum}{4}$对“较好的”分块，因为每对分块都有两个分块并且每对分块又会扩展成两个不同的分块对。考虑到各分块大小不尽相同，选取“较好的”分块是用相似重复数据存在率 e 来评价，计算公式如下：

$$e_{nm} = \frac{dn_{nm}}{bc_{nm}} \tag{4}$$

式中：e_{nm} 表示 n 分块和 m 分块间的相似重复数据存在率，dn_{nm} 表示 n 分块和 m 分块间的相似重复数据，bc_{nm} 表示 n 分块和 m 分块间的数据比较次数。

(3) 滑动窗口选取：

分块近邻匹配重复检测算法(BNM)使用固定窗口，窗口大小的选取一直存在问题。窗口选择过小会使得检验的结果不准确，窗口选择过大会增加不必要的比较。新的算法(MBNM)使用自适应的滑动窗口。

首先，设置三个参数：窗口最小值、窗口最大值、阈值。本文定义 $w_{\min} = 4$。窗口初始值 $w = w_{\min}$ ($w_{\min} \leqslant w \leqslant w_{\max}$)。如果两个记录的相似度大于阈值，则表示对应窗口内的排序足够好。

(4) 分块比较：

算法 3 比较算法

compare(blocks,bPairs,order)

输入：待处理的数据集 D 和排序字段集 Ks

输出：比较结果 bPairs

```
(1) for(k = 0;k <= |Ks| - 1;k ++){
(2)     bPairs = { <1,1, -1,k>,⋯, <bNum,bNum, -1,k> }
(3)     order[k] = sort(D,Ks[k],S,bPairs)
(4)     bPairs = bPairs ∪ pairs
(5) }
(6) blocks = loadblocks(bPairs,S,order)
(7) compare(blocks,bPairs,order)
(8) bPairs = bPairs ∪ pBPs
```

blocks 集合保存着分块的数据；bPairs 集合用于存放分块对信息，分块对信息采用一个三元组(分块 1 编号，分块 2 编号，相似重复记录对数)来表示。分块之间对比形成如图 1 的分块矩阵，分块矩阵存在对称性，

因此只分析行列式的右上部分。采用二元组(b_n,b_m)表示分块之间的数据对比,相似重复记录用(r_i,r_j)表,C_{nm}表示分块 n 和分块 m 之间存在相似重复数。

$$(b_n,b_m)=\{(r_i,r_j)\mid r_i\in b_n,r_j\in b_m\} \tag{5}$$

$$|(b_n,b_m)|=dn_{nm} \tag{6}$$

先计算各分块内的相似重复数,得到初始分块对相似重复数据,即(b_n,b_n),选取 dn_{nn} 较大的分块对(b_n,b_n)并向旁边分块扩展生成新分块对,记作$(b_n,b_m)^+$。

$$(b_n,b_m)^+=\{(b_{n+1},b_m),(b_n,b_{m+1})\} \tag{7}$$

为了确保分块对能有较高的相似重复数据,设置分块间隙 R 限制分块对的扩展,减少比较次数。$|n+1-m|\leqslant R\&|m+1-n|\leqslant R$。

对比新分块对得到各分块对的相似重复记录数。重复上述过程直到所有的分块对比对过。

(5) 计算传递闭包:

由于成对的记录的选取比对,导致运行的结果的不是关闭的,例如(r_a,r_b)和(r_b,r_c)被检测出是重复记录,但是最后的运行结果可能没有(r_a,r_c)。因此,在算法最后中用传递闭包技术不仅能得到较好的重复记录集,并能解决部分纰漏的问题。

3.3　算法复杂度分析

多字段分块近邻匹配算法得时间复杂度主要包括三部分,记作 T_1、T_2、T_3。T_1表示多关键字选取的时间复杂度,根据检测目标来确定数据集 D_K,$T_1=O(N)$;算法进行多次排序,排序后调用 *bPairs* 函数,因为 D_K 的个数不大于 k,$T_2\leqslant(N\log N+2mw)$,$m<N$,$w<N$。其中 m 为窗口数,w 为滑动窗口大小,$T_2=O(N\log N)$。比较的时间复杂度为$[n((m/4)+(w/2))]$。其中 n 是每个记录中的字段数,当 $n<\log N$,$T_3=O(n)$。当 $n>\log N$,$T_3=O(N\log N)$。综上所述,MBNM 算法的时间复杂度为 $\Theta(n((m+2w)/4+\log n))$。其他算法时间复杂度[3]如表 1 所示。

表 1　时间复杂度比较

算法	比较次数	匹配时间复杂度	排序时间复杂度	总计
SBM	$(p-1)(u^2+u)/2+p(m^2-m)/2$	$O(nm/2)$	$O(n\log n)$	$\Theta(n(m/2+\log n))$
MPN	$k(w-1)(n-w/2)$	$O(nkw)$	$O(n\log n)$	$\Theta(n(kw+\log n))$
BNM	$[n(s-1)+s(w-1)(2n-sw)]/2$	$O(ns/2)$	$O(n\log n)$	$\Theta(n((2ws-s-1)/2+\log n))$
MBNM	$[n((m/4)+(w/2))]$	$O((nm+2nw)/4)$	$O(n\log n)$	$\Theta(n((m+2w)/4+\log n))$

改进算法根据不同的关键字段排序,为了节约内存,改进算法只保存排序后的数据的主键。改进算法在执行时优先选择相似重复率较高分块延展,降低了排序字段本身好坏对算法运行时间的影响。改进算法放弃延展过慢的分块,进一步减少算法中无意义的数据比较,从而提高算法的运行效率。

4　实验结果分析

4.1　实验环境

实验环境是 Lenovo PC CPU Inter(R)Core(TM)2 Duo E8400 @ 3.00 GHz,Ram 2.00 GB,Windows 7 32 位操作系统。采用 Eclipse Java 编程工具实现算法,Java 环境 Java 1.7 以上。

实验数据来源于某市国家电网用户信息的采集数据,包括用户的用电数据、部分试点事业单位的采集数据等,由于来源广泛导致采集到的数据必然存在大量的重复。度量相似检测算法有效性的三个主要标准包括查全率、查准率和 F 值,评价算法复杂度的参考标准为运行时间。为了检验检测算法的有效性,设计以下实验。分别从数据源中提取四组数据,对四种算法进行比较,四组数据量分别为 58.3、95.1、128.8 和 136.7 万。通过软件和人工等方式对上数据分别处理,使之分别包含 0.43、0.87、1.38 和 1.72 万条相似重复记录。对数据集观察分析后,确定使用其中的“name”、“contacts”、“title”三个字段为关键排序字段。

评价标准采用了查全率(Recall)、查准率(Precision)、F 值(F-score)和运行时间四个指标来比较四种算法。查全率,查准率,F 值计算公式如下:

(1) 查准率:

$$precision=\frac{TP}{TP+FP} \tag{8}$$

(2) 查全率:

$$recall=\frac{TP}{TP+FN} \tag{9}$$

(3) F 值(F-score):

$$F\text{-}score=\frac{2\times precision\times recall}{precision+recall} \tag{10}$$

其中 TP 为被正确识别的重复元组数,FP 为错误识别出的重复元组数,FN 未识别出的重复记录数。则 $TP+FP$ 为识别出的重复元组数,包括正确识别的和错误识别的重复元组数。$TP+FN$ 为实际存在的重复

元组总数。

4.2 相似重复检测算法对比

MBNM 算法使用的多关键字排序和自适应滑动窗口来进行数据比较前的处理。由于分块大小和分块间隙对算法效率有直接影响。在保证四种不同算法的分析数据量相同的情况进行了比较实验。四种算法有着明显的差别,比较结果如下:

实验表明,SBM 由于只采用传统的固定窗口,识别的结果受窗口大小影响较大。MPN 算法由于固定滑动窗口,窗口选取过大或过小对查全率影响较大。由于 MBNM 采用自适应滑动窗口和多趟检测技术,大大减少记录比对时间和总体测时间,既保证查全率又在一定程度上减少检测时间,从而提高了查全率。如图 2 所示。

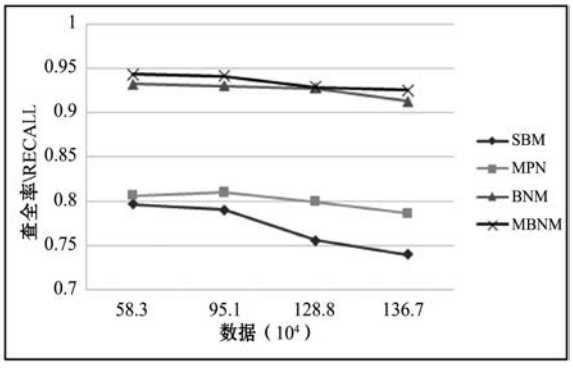

图 2 查全率比较图

实验表明,MPN 算法由于采用传递闭包,准确率明显较低。在数据量较小时,四者的查准率差异较小,但随着数量的增加,四者查准率都有一定程度的降低,最终趋于稳定。MBNM 算法的查准率更加稳定且查准率较高。如图 3 所示。

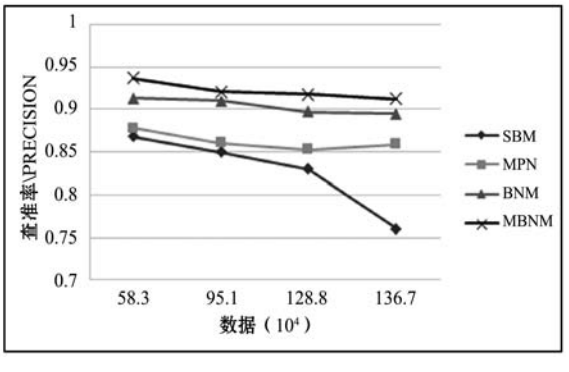

图 3 查准率比较图

MPN、SBM、BNM、MBNM 四种算法的 F 值与数据量之间的关系如图 4 可以看出。算法的 F 值随着数据量的增加而降低,并且四种算法的 F 值之间的差异越来越大,MBNM 略优于 BNM 算法。

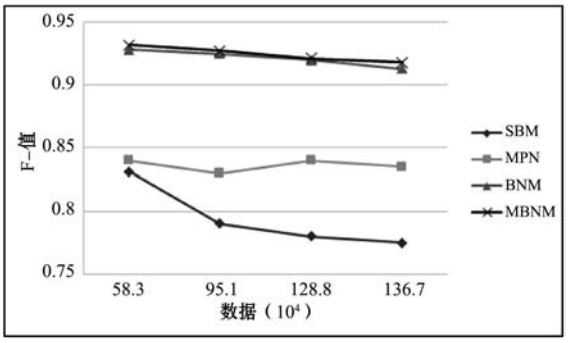

图 4 F 值比较图

图 5 比较结果表明,在相同机器、相同数据下,MBNM 算法在时间复杂度上有较好的表现。由于利用自适应滑动窗口降低了比较数据集,并利用 K 关键字多次排序一次执行加快了数据检测的效率。本文提出的算法的时间开销与相似重复记录的多少有关,相似重复记录数越多,则时间复杂度越低,反之越高。

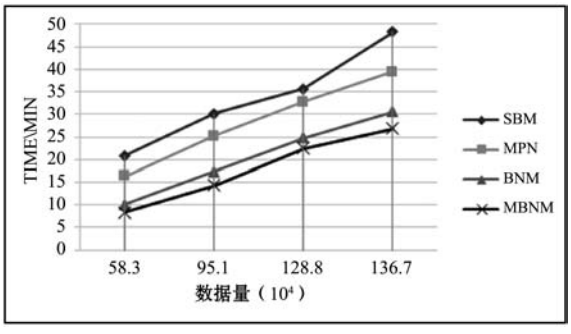

图 5 算法消耗时间比较图

综上所述,本文提出的多排序字段分块近邻匹配算法(MBNM)的运行效率优于其他算法。

5 结 语

本文研究了如何检测重复记录,结合了窗口技术和分块技术,提出了分块近邻匹配算法,该算法在限定的时间内提高了相似重复记录匹配的效率。算法通过实时的分块匹配结果动态地改变了匹配数据的顺序,优先对更有希望是重复的数据对进行匹配,并舍弃较差的分块对的比较。减少了算法的比较次数进而减少了算法的时间,并提出了多字段排序和自适应滑动窗口技术改进算法,多个字段同时和自适应滑窗进行分块比较。最后并通过实验和传统“排序 & 合并”思想中的其他两种算法比较验证了该算法的有效性。接下来的工作就是优化算法的某些过程,进一步缩短算法的运行时间和提高算法的效率并应用到实际中去。

参 考 文 献

[1] 魏晓菁, 陈峰, 董媛媛. 数据资产可信度评估模型研究[J]. 计算机应用, 2015(S2):170-173.

[2] Draisbach U, Naumann F. A Comparison and Generalization of Blocking and Windowing Algorithms for Duplicate Detection[C]//Proceedings of International Workshop on Quality in Databases, 2009:51-56.
[3] Bilenko M: Adaptive blocking: Learning to scale up record linkage [C]//International Conference on Data Mining. IEEE Computer Society, 2006:87-96.
[4] Hernández M A, Stolfo S J. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem[J]. Data Mining & Knowledge Discovery, 1998, 2(1):9-37.
[5] Papenbrock T, Heise A, Naumann F. Progressive Duplicate Detection[J]. IEEE Transactions on Knowledge & Data Engineering, 2015, 27(5):1316-1329.
[6] Ma M, Wang P, Chu C H, et al. Efficient Multipattern Event Processing Over High-Speed Train Data Streams[J]. IEEE Internet of Things Journal, 2015, 2(4):295-309.
[7] Yu Z, Kuang Z, Liu J, et al. Adaptive ensembling of semi-supervised clustering solutions[J]. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(8): 1577-1590.
[8] 郑津杨, 徐坤, 李建强. 用于 RFID 系统数据处理的排序邻居算法性能分析[J]. 计算机应用与软件, 2016, 33(12):207-210.
[9] 肖满生, 周浩慧, 王宏. 基于模糊综合评判的相似重复记录识别方法[J]. 计算机工程, 2010, 36(13):51-53.
[10] 刘雅思, 程力, 李晓. 基于长度过滤和动态容错的 SNM 改进算法[J]. 计算机应用研究, 2017, 34(1): 147-150.
[11] 杨巧巧, 郭振波, 王开西. 基于聚类分组和属性综合权值的 SNM 改进算法[J]. 工业控制计算机, 2017(9):27-28.
[12] Draisbach U, Naumann F, Szott S, et al. Adaptive Windows for Duplicate Detection[C]//IEEE International Conference on Data Engineering. IEEE, 2012:1073-1083.
[13] Christen P. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication[J]. IEEE Transactions on Knowledge & Data Engineering, 2011, 24(9): 1537-1555.
[14] Subramaniyaswamy V, Pandian S C. A Complete Survey of Duplicate Record Detection Using Data Mining Techniques [J]. Information Technology Journal,2012,11(8):941-945.
[15] 陈爽, 刁兴春, 宋金玉,等. 基于伸缩窗口和等级调整的 SNM 改进方法[J]. 计算机应用研究, 2013, 30(9):2736-2739.
[16] 邱越峰, 田增平, 季文赟,等. 一种高效的检测相似重复记录的方法[J]. 计算机学报, 2001, 24(1):69-77.
[17] 宋人杰, 余通, 陈宇红,等. 基于 MapReduce 模型的大数据相似重复记录检测算法[J]. 上海交通大学学报, 2018, 52(2):214-221.

(上接第 246 页)

[4] Nguyen T H, Grishman R. Relation Extraction: Perspective from Convolutional Neural Networks[C]//The Workshop on Vector Space Modeling for Natural Language Processing, 2015:39-48.
[5] Santos C N D, Xiang B, Zhou B. Classifying relations by ranking with convolutional neural networks [J]. Computer Science, 2015, 86(86):132-137.
[6] Zhang D, Wang D. Relation classification via recurrent neural network[EB]. arXiv:1508.01006, 2015.
[7] Zeng D, Liu K, Chen Y, et al. Distant supervision for relation extraction via piecewise convolutional neural networks [C]//Conference on Empirical Methods in Natural Language Processing. 2015:1753-1762.
[8] Lin Y, Shen S, Liu Z, et al. Neural relation extraction with selective attention over instances[C]//Meeting of the Association for Computational Linguistics,2016:2124-2133.
[9] Kim Y. Convolutional neural networks for sentence classification[C]//2017 XLIII Latin American Computer Conference (CLEI). IEEE, 2014: 1746-1751.
[10] Liu Y, Wei F, Li S, et al. A dependency-based neural network for relation classification[EB]. arXiv: 1507.04646, 2015.
[11] Manning C D, Surdeanu M, Bauer J, et al. The Stanford CoreNLP Natural Language Processing Toolkit [C]//Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations,2014.

(上接第 254 页)

[3] 卢凯,徐建闽,叶瑞敏. 经典干道协调控制信号配时数解算法的改进[J]. 公路交通科技,2009,26(1): 120-124,129.
[4] 栗红强. 城市交通控制信号配时参数优化方法研究[D]. 长春:吉林大学,2004.
[5] Little J D C, Kelson M D, Gartner N M. MAXBAND: A program for setting signals on arteries and triangular networks [J]. Transportation Research Record, 1981, 795:40-46.
[6] Gazis D C. Traffic theory[M]. New York: Springer,2002.
[7] 何尚秋,郭海锋,俞立,等. 基于剪枝法的交通信号相位优化设计[C]//第三届国际电力电子与智能交通会议. 2010:425-428.
[8] 中华人民共和国公安部. GB/T 31418-2015 道路交通信号控制系统术语[S]. 北京:中国标准出版社,2015.
[9] Lu K, Zeng X, Li L, et al. Two-Way Bandwidth Maximization Model with Proration Impact Factor for Unbalanced Bandwidth Demands[J]. Journal of Transportation Engineering, 2012, 138(5):527-534.
[10] 王江静,姜久雷. 可伸缩矢量图形(SVG)[J]. 现代电子技术, 2005, 28 (24):32-33.