软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(6):2331−2347 [doi: 10.13328/j.cnki.jos.006284]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向查询式实体解析的多属性数据索引技术*

孙琛琛[1], 申德荣[2], 肖迎元[1], 李玉坤[1]

[1](计算机视觉与系统省部共建教育部重点实验室(天津理工大学), 天津 300384)

[2](东北大学 计算机科学与工程学院, 辽宁 沈阳 110169)

通信作者: 孙琛琛, E-mail: suncc@email.tjut.edu.cn

摘 要: 实体解析是数据集成的关键方面, 也是大数据分析与挖掘的必要预处理步骤. 大数据时代, 随着查询驱动的数据应用需求的不断增长, 查询式实体解析成为热点问题. 为了提升查询-解析效率, 研究了面向实体缓存的多属性数据索引技术. 涉及两个核心问题: (1) 如何设计多属性数据索引? 设计了基于R-树的多属性索引结构. 为了满足实体缓存在线生成需求, 提出了基于空间聚类的在线索引构建方法. 提出了基于"过滤-验证"的多维查询方法, 利用多属性索引有效地过滤掉不可能命中的记录, 然后采用相似性函数或距离函数逐一验证候选记录. (2) 如何将不同的字符串属性插入到树形索引中? 解决思路是, 将字符串映射到数值空间. 针对Jaccard相似性和编辑相似性, 提出了基于 *q*-gram 的映射方法, 并提出了基于向量降维的优化和基于 *z*-order 的优化, 实现高质量的"字符串→数值"映射. 最后, 在两个数据集上进行实验评估, 验证多属性索引的有效性, 并测试其各个方面.

关键词: 实体解析; 多属性数据索引;查询式; 数据集成; 数据预处理

中图法分类号: TP311

中文引用格式: 孙琛琛, 申德荣, 肖迎元, 李玉坤. 面向查询式实体解析的多属性数据索引技术. 软件学报, 2022, 33(6): 2331−2347. http://www.jos.org.cn/1000-9825/6284.htm

英文引用格式: Sun CC, Shen DR, Xiao YY, Li YK. Multi-attribute Data Indexing for Query Based Entity Resolution. Ruan Jian Xue Bao/Journal of Software, 2022, 33(6): 2331−2347 (in Chinese). http://www.jos.org.cn/1000-9825/6284.htm

Multi-attribute Data Indexing for Query Based Entity Resolution

SUN Chen-Chen[1], SHEN De-Rong[2], XIAO Ying-Yuan[1], LI Yu-Kun[1]

[1](Key Laboratory of Computer Vision and System of Ministry of Education (Tianjin University of Technology), Tianjin 300384, China)

[2](School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China)

Abstract: Entity resolution is a key aspect of data integration, and also is a necessary preprocessing step of big data analytics and mining. In big data era, more and more query-driven data analytics applications come out, and query-based entity resolution becomes a hot topic. This work studies multi-attribute data indexing technology for entity cache in order to promote query-resolution efficiency. There are two core problems. One is how to design the multi-attributeindex. An R-tree based multi-attributeindex is designed. Entity cache is produced online, so an online index construction method is proposed based on spatial clustering. A filter-verify based multi-dimensional query method is proposed. It filters impossible records by the multi-attributeindex, and then verifies each candidate record with similarity functions or distance functions. The other ishow to insert different string attributes into the tree index. The basic solution is mapping strings into integer spaces. For Jaccard similarity and edit similarity, a q-gram based mapping method is proposed, and is improved by vector dimension reduction and z-order, which achieves high mapping qualities. Finally, the proposed hybrid index is experimentally evaluated on two datasets. Its effectiveness is validated, and moreover, different aspects of the multi-attribute index are also tested.

Key words: entity resolution; multi-attribute data indexing; query based; data integration; data preprocessing

实体解析(entity resolution, ER)是数据集成和数据预处理的重要环节, 也是大数据分析与挖掘的基础工

* 基金项目: 国家自然科学基金(62002262, 61672142, 61602103, 62072086, 62072084); 国家重点研发计划(2018YFB1003404)
收稿时间: 2020-08-02; 修改时间: 2020-09-09, 2020-11-20; 采用时间: 2020-12-16; jos 在线出版时间: 2021-04-20

作[1,2]. 实体解析致力于找出数据源中描述相同实体的不同数据记录, 也被称为实体匹配和记录链接. 传统的实体解析是批处理式任务, 直接解析整个数据集, 因此时间开销较大. 然而在大数据时代, 面对大规模的数据量, 查询驱动的数据分析应用成为热点需求, 为此需要提供查询式实体解析服务. 不同于批处理式实体解析, 查询式实体解析需要近似实时地完成面向查询记录的查询-解析任务[3].

当前, 查询式实体解析有着广泛的应用前景. 比如, 反恐信息系统收集来自多个数据源的不同方面的个人信息, 包括公安居民信息数据库、出入境信息数据库、犯罪信息数据库、火车票务数据库以及移民信息数据库等. 当进行嫌疑恐怖分子信息定位时, 需要从这些数据源中查询-解析出关于该嫌疑恐怖分子的所有数据记录, 从而协助案情侦测. 再比如, 小型公司获得较大量的数据, 然而只关心一小部分与自己商业运营有关的数据, 因此只需要查询-解析出自己感兴趣的数据记录即可.

查询式实体解析中近似实时的效率需求离不开有效的实体缓存机制, 而实体缓存的核心在于高效的索引技术. 数据记录包含多个属性, 以字符串为主, 还有少量数字串, 并且不同属性通过不同类型的相似性(比如 Jaccard、编辑距离、数值的差值等)衡量, 这就构成多属性多相似性空间——一种特殊的多维空间, 每个维度对应特定的属性类型和相似性类型. 实体缓存需要支持多属性多相似性空间搜索, 因此需要设计面向多属性多相似性空间的索引, 称为多属性数据索引. 然而, 已有的字符串索引大部分是一维的[4], Zhang 等人提出了基于编辑距离的字符串索引[5], Zhang 等人提出了基于集合相似性的字符串索引[6]. Li 等人提出了基于前缀树的多属性相似性连接与查询方法, 可以支持多维的字符串查询, 需要构建多个前缀树[7]. 多维索引技术则主要面向多维数值空间[8]. 总体而言, 当前缺乏高效的面向多属性多相似性空间的数据索引技术.

本文提出了基于 R-树的多属性数据索引技术来解决实体缓存查询问题. R-树是 B-树在多维空间的拓展, 解决了多维空间搜索问题[9], 因此以 R-树为基础来设计多属性索引结构. 然而, R-树面向数值, 无法处理字符串. 为此, 将字符串映射到数值空间, 以便将其插入到树形索引. 给定字符串相似性函数, “字符串→数值”映射必须满足相似性上界可计算, 即线性时间内计算出查询字符串与一个字符串范围的最大相似性, 从而实现有效的数据过滤. 以 Jaccard 相似性和编辑相似性为例, 设计了基于 q-gram 的映射方法, 并且通过向量降维和 z-order 变换来优化之, 有效提升了过滤能力. 在此基础上, 提出了多维查询方法, 先通过多属性索引过滤掉不可能命中的记录, 然后采用相似性函数来验证候选记录是否满足查询约束. 查询式实体解析中, 随着查询-解析的进行, 缓存记录在线生成, 为此设计了小批量式在线索引构建方法, 保证实时性. 通过实验评估表明了所提出的多属性索引技术的有效性. 需要说明的是, 本文主要针对全局数据不变的场景展开研究.

本文的主要贡献如下:

- 提出了基于 R-树的多属性数据索引技术. 设计了基于 R-树的多属性索引结构; 提出了基于空间聚类的小批量式在线索引构建方法; 提出了基于“过滤-验证(filter-verify)”的多维查询方法;
- 提出了一组“字符串→数值”映射方法. 首先提出了基于 q-gram 的基本映射方法; 然后提出了两个基于降维的优化方法: 基于哈希的降维和基于频率分类的降维; 最后提出了基于 z-order 的优化方法. 其中, 基于哈希的降维方法可用于全局数据变化的场景, 而基于频率分类的降维方法只适用于全局数据不变的场景;
- 在两个数据集上进行了充分的实验评价, 通过与已有工作的对比, 验证了本文提出的索引技术的有效性, 并且测试了索引技术自身的各个方面.

本文第 1 节进行研究概述, 包括查询式实体解析、实体缓存机制和问题定义. 第 2 节介绍基于 R-树的多属性数据索引, 包括索引结构、索引构建和多维查询. 第 3 节详述“字符串→数值”映射, 包括基于 q-gram 的映射和两类优化方法: 向量降维和 z-order 变换. 第 4 节进行实验评价, 将本文提出的多属性索引技术与已有工作对比, 并且测试多属性索引的各个方面. 第 5 节回顾相关工作, 涉及查询式实体解析、字符串索引技术和空间索引技术等. 最后, 第 6 节对全文进行总结, 并指出下一步可能的研究方向.

1 研究概述

1.1 查询式实体解析

如图1所示, 给定一条查询记录 r_q 和脏数据集 D, 从 D 中找出所有与 r_q 匹配的记录并组成记录类簇 R_q, 将 R_q 作为查询式实体解析的结果返回. 为了保证响应速度和用户体验, 需要在较短时间内返回精确的查询-解析结果.

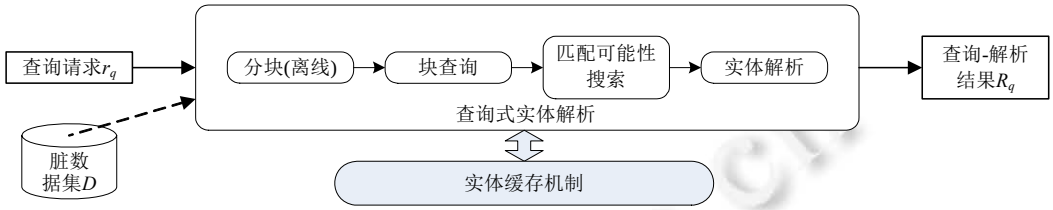


图 1 查询式实体解析架构

查询式实体解析(query based entity resolution, Q-ER)采用"过滤-验证"策略.

(1) 分块(离线)与块查询. 离线时, 对脏数据集进行多路分块, 得到块集合[10,11]. 查询时, 利用查询记录生成分块键值(blocking key value, BKV)[10,11], 进而查询得到该记录对应的块, 这些命中的块中所有记录构成初步的候选记录集合. 在此过程中, 利用开销较小的分块技术过滤大部分不可能匹配的数据记录;

(2) 匹配可能性搜索. 利用分块来计算查询记录与候选记录的匹配可能性[2], 并选择 top-K 个记录作为最终候选记录;

(3) 实体解析. 根据匹配可能性, 对查询记录与候选记录依次进行实体解析, 所有匹配的记录组成一个记录类簇[12], 返回记录类簇. 如果返回的记录类簇不为空, 则查询-解析命中; 否则, 查询-解析未命中. 一个实体解析方法 $m(*,*)$接受两个记录作为输入, 输出二者是否匹配.

一方面, 实体解析的时间开销较大, 并且候选记录对的数量较多, 因此导致查询-解析的时间开销较大, 影响查询-解析响应速度; 另一方面, 已有的查询-解析结果具有复用价值. 为了提升查询-解析效率, 应设计高效的缓存机制来协助 Q-ER.

1.2 面向实体解析的缓存机制

实体缓存收集已有的查询-解析结果, 以便后续相似的查询到来时直接返回结果. 记录类簇(即查询-解析结果)包含多条记录, 通过数据融合[13,14]生成一条代表记录来代表该类簇[15], 缓存操作将基于代表记录. 数据融合方法[13]为每个属性选择最有代表性的值, 这样的值包括了类簇中该属性对应的值的所有信息, 并且消除掉数据冲突, 保证了当前匹配的正确性和后续匹配的正确性, 这样最大限度地提高了实体解析的准确性[15]. 如算法 1 所示, 实体缓存机制的基本工作原理如下: 当查询请求 r_q 到来时, 首先进行缓存查询(第 1 行), 如果命中, 则依次将查询记录与命中的缓存记录进行实体解析: 若发现匹配记录, 则直接返回匹配记录类簇(第 2–5 行); 如果缓存查询未命中, 则执行标准的查询式实体解析, 得到查询-解析结果 R_q(第 6 行), 若 R_q 不为空, 则将其加入到实体缓存(第 7–9 行). 其中, $EC.query(r_q)$对缓存 EC 进行查询, 返回命中的候选记录集合 R_c; $m(*,*)$对两记录进行实体解析, 如果匹配, 返回真, 否则, 返回假; $EC.recordCluster(r_c)$返回缓存中 r_c 对应的记录类簇; $dataFustion(R_q)$生成代表记录 r_c, 为每个属性生成最有代表性的值; $EC.insert(r_c,R_q)$将当前查询-解析结果的代表记录 r_c 及其记录类簇 R_q 加入缓存.

算法 1. 基于实体缓存的 Q-ER.

输入: 查询记录 r_q, 脏数据集 D, 实体缓存 EC;

输出: 查询-解析结果 R_q.

```
1.  Rc=EC.query(rq);                          //实体缓存查询
2.  if Rc!=Φ
3.     for rc∈Rc
4.        if m(rq,rc)                         //实体解析
5.           return EC.recordCluster(rc);     //缓存中至多存在一个与查询记录匹配的缓存记录 rc
6.  Rq=Q-ER(rq,D);                            //标准的查询式实体解析方法
7.  if Rq!=Φ
8.     rc=dataFustion(Rq);                    //数据融合
9.     EC.insert(rc,Rq);                      //将最新的查询-解析数据加入实体缓存
10. return Rq;
```

实体缓存机制的核心问题之一是如何实现高效、准确的记录查询. 实体解析中, 数据记录通常包含多个不同特点的字符串属性及数值属性, 不同的字符串属性可能对应不同的字符串相似性函数和阈值, 比如姓名对应编辑距离、工作单位对应 Jaccard 相似性、年份对应差值, 阈值分别为 2、0.75 和 1. 已有的多维索引主要面向数值型数据, 字符串索引通常是一维的, 都无法满足上述需求. 为此, 需要设计支持多维查询的索引, 这是本文的研究主题.

1.3 问题定义

数据记录集合 $S=\{s_i|0\leqslant i\leqslant n\}$, 对应的属性集合为$\{S^j|0\leqslant j\leqslant x\}$; 记录 s_i 的属性值集合为 $\{s_i^j\,|\,0\leqslant j\leqslant x\}$, s_i^j 表示记录 s_i 在属性 S^j 上的值. 不同类型的属性需要通过不同类型的方法来衡量相似性, 字符串属性通过相似性函数衡量, 数值属性通过距离函数衡量. 字符串相似性函数比较复杂、多样, 而数值的距离计算相对简单(比如差值). 实体解析中用到各种字符串相似性函数, 可分为两类.

(1) 基于编辑的相似性: 通过衡量两个字符串之间的字符级变换来计算相似性, 也称为基于字符(character)的相似性. 以编辑距离(edit distance, ED)为例, $ED(r_q^j,s_i^j)$ 是将 r_q^j 通过单个字符的增加、删除和替换这 3 个基本操作转换成 s_i^j 所需要的最少操作次数. 为了方便使用, 可将编辑距离转换成编辑相似性(edit similarity, ES):

$$ES(r_q^j,s_i^j)=1-\frac{ED(r_q^j,s_i^j)}{\max\{|\,r_q^j\,|,|\,s_i^j\,|\}} \tag{1}$$

其中, |*|表示字符串长度. 同类型的相似性还有 Jaro、Jaro-Winkler 等;

(2) 基于集合的相似性: 将字符串处理成标识集合, 然后利用集合相似性函数来计算字符串相似性, 也称为基于标识(token)的相似性. 以 Jaccard 为例:

$$Jaccard(r_q^j,s_i^j)=\frac{|\,tokenize(r_q^j)\cap tokenize(s_i^j)\,|}{|\,tokenize(r_q^j)\cup tokenize(s_i^j)\,|} \tag{2}$$

其中, *tokenize*(*)是标识生成函数. 本文提到的标识是 *q*-gram 级别的, 比单词级标识的粒度更细, 计算得到的相似性更准确且容错性更强. 同类型的相似性还有 Overlap、Cosine 等.

定义 1(多维查询). 给定数据集 S 和查询记录 r_q, 二者属性相同, 多维查询约束Φ如下所示:

$$\Phi=\wedge_{j\in QCA_{sim}}\varphi(S^j,r_q^j,sim^j,\theta^j)\wedge_{k\in QCA_{dist}}\varphi(S^k,r_q^k,dist^k,\theta^k) \tag{3}$$

其中, $QCA=QCA_{sim}\cup QCA_{dist}$, 是查询约束属性编号集合.

属性相似性衡量方式分为两类, 对应两类约束: $\varphi(S^j,r_q^j,sim^j,\theta^j)=\{sim^j(S^j,r_q^j)\geqslant\theta^j\}$, sim^j 为相似性函数, $\varphi(S^k,r_q^k,dist^k,\theta^k)=\{dist^k(S^k,r_q^k)\leqslant\theta^k\}$, $dist^k$ 为距离函数. 那么, 多维查询返回符合Φ的数据记录.

将采用“过滤-验证”策略来解决面向数据记录的多维查询问题: (1) 过滤阶段, 通过多属性索引过滤掉不可能满足查询约束的记录, 生成候选记录集合; (2) 验证阶段, 调用各种相似性函数或距离函数计算候选记录

是否满足查询约束. 查询的效率和精确性主要依赖于多属性索引的过滤能力, 为此, 将研究基于 R-树的多属性索引.

2 基于 R-树的多属性数据索引

实体缓存索引应满足以下要求: ① 支持多维多类型(以字符串和数值为主)属性; ② 针对不同的属性, 支持不同的相似性函数, 如编辑相似性或 Jaccard 相似性; ③ 支持多维查询, 每个维度上满足各自阈值的约束; ④ 支持查询时在各个维度自由地设定阈值, 而不必在构建索引时就固定阈值. 为此, 设计了基于 R-树的多属性数据索引.

2.1 多属性索引的基本思想

R-树是一种高效的多维索引, 支持快速的多维查询、更新代价较小、支持外存, 但只适用于数值, 而不适用于字符串, 因此将改进 R 树以设计多属性索引. 如例 1 所示, 多维树形索引中, 每个非叶子节点包含多个成员, 每个成员包含一个最小临界矩形(minimum bounding rectangle, MBR)和指向孩子节点的指针; 叶子节点存储一组数据记录. MBR 是多维索引的核心, 它表示多维空间中一块区域, 在每个维度上限定一个取值范围. 其中, 对于 R 树中根节点以外的节点来说, 其节点中数据量满足范围约束[*m*,*M*]. 字符串本身不是可度量的, 因此无法对字符串直接构建索引. 解决思路是: 将字符串基于给定相似性函数映射到数值空间, 将对字符串的索引转变为对数值的索引. 例 1 中, 将名字和职业分别映射成二进制的数字, MBR 中字符串的取值范围就可以用二进制取值范围表示. 给定相似性函数, 两个字符串的相似性将通过它们的对应数值来计算. 显然, "字符串→数值"映射需要满足某些要求.

例 1: 如图 2 所示, 针对个人信息记录中名字、职业和出生年份这 3 个属性构建三维索引, 名字通过编辑相似性衡量, 职业通过 Jaccard 相似性衡量, 出生年份的距离函数为差值. 其中, 名字和职业两个字符串属性被映射成二进制数字.

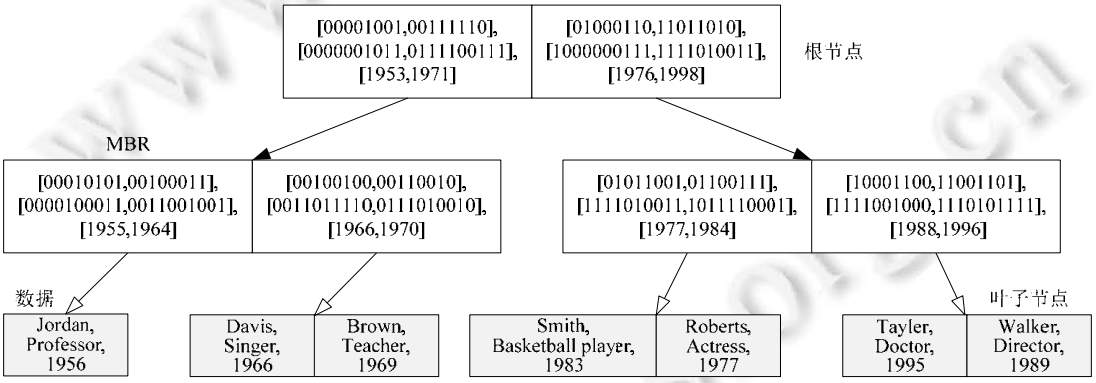


图 2 高度为 3 的三维索引

定义 2("字符串→数值"映射). 给定一个字符串集合 T, 存在映射函数ξ:$T\to N$, 将字符串集合中每个字符串映射到唯一数值.

性质 1(可比较性). 给定任意两个字符串 t_i 和 t_j, $\xi(t_i)$与$\xi(t_j)$的大小关系唯一确定, 这是一种全序关系.

性质 2(相似性上界可计算). 给定相似性函数 $sim(*,*)$、查询字符串 t_q 和字符串范围$[t_i,t_j]_\xi$, 可在线性时间内计算出 $\max(sim(t_q,t_k))$, $t_k\in[t_i,t_j]_\xi$.

命题 1. 给定相似性函数 $sim(*,*)$, 当且仅当"字符串→数值"映射ξ满足性质 1 和性质 2 时, 映射ξ可以用于构建多属性索引.

多属性索引的插入和删除操作都依赖于可比较性, MBR 每个维度都必须具有可比较性. 相似性上界计算

的效率和有效性将决定索引的过滤能力. 一方面, 基于树形索引的多维查询是一个自顶向下的过程, 在每一层都要执行多次相似性上界计算, 因此上界计算的效率将影响多维查询的效率; 另一方面, 如果相似性上界计算不准确, 可能导致过度调用字符串相似性函数 *sim*(*,*)来验证, 字符串相似性函数的开销通常较大, 从而产生巨大的开销, 严重影响到索引的过滤能力. 为了确保多维查询的效率, 映射应该保证高效(即线性时间内)、准确的相似性上界计算.

2.2 多属性索引在线构建

查询式实体解析中查询-解析结果在线地生成, 需要近似实时地加入缓存. 传统的R-树构建过程包含频繁的插入和分裂操作[9], 导致时间开销较大, 应当考虑批量插入. 已有的批量插入方法要么静态地从 0 开始生成完整的 R-树[16,17], 要么动态地将(较大量的)增量数据一次性插入到已有 R-树[18,19], 不适用于实体缓存的在线生成需求. 为此, 采用批-流融合的理念, 每次处理小批量的增量数据, 即数据增量不超过 500. 提出了一种基于空间聚类的在线索引构建方法. 为了构建多属性索引, 首先需要将记录转换成多维数值空间中的点, 然后将点插入到树形索引.

多属性索引在线构建的基本思想是: 将增量数据进行聚类, 以类簇为单位进行 best-effort 插入. 如算法 2 所示, 多属性索引在线构建流程如下.

- 将数据记录转换为多维数值空间内的点(第 1 行), 数据空间的维度取决于多属性索引的维度$|IA|$≤记录的属性维度;
- 利用 GDBSCAN 算法[20]对多维数值型数据Γ进行基于密度的空间聚类(第 2 行), 得到类簇集合 *CL*;
- 从 *CL* 中迭代地选择最大类簇 *cl*, 执行以下操作(第 6–10 行): 利用 *cl* 生成 MBR *mbr*(第 7 行); 然后调用 *insert* 函数, 将 *mbr* 中的数据尽量多地插入到 *MRTree* 的某个叶子节点, *cl'*为无法插入的数据集合(第 8 行);
- 当遍历完 *CL* 后, 若还有剩余数据, 则对剩余数据进行空间聚类(第 11 行、第 12 行);
- 循环执行上述操作(第 4–14 行), 直到 *CL* 为空.

需要说明的是: 在迭代过程, 一旦被 GDBSCAN 算法判别为离群点数据, 后续就不会被 GDBSCAN 算法再次处理(第 3 行、第 13 行), 而是最后被逐个插入到 *MRTree*(第 15 行、第 16 行).

insert 函数是对传统 R-树中插入操作[9]的改进, 它的输入为一个 MBR *mbr*. 首先, 找出 *mbr* 的中心点在 *MRTree* 中对应的叶子节点 *LF*(第 1 行); 接着, 将 *mbr* 中符合 *LF* 的 MBR 约束的数据插入到 *LF* 中(第 2 行); 然后, 在必要情况下, 执行节点分裂、向上传递、树增高等 R-树基本操作(第 3–5 行)[9]; 最后, 返回剩余的数据 *cl*(第 6 行).

算法 2. 多属性索引在线构建.

输入: 新增(小批量)数据记录集合Δ, 多属性索引 *MRTree*, 索引属性编号集合 *IA*, 映射集合$\{\xi_j|j\in IA\}$;

输出: 更新后的多属性索引 *MRTree*.

1. Γ=*Record2IntegerPoint*(Δ,*IA*,$\{\xi_j|j\in IA\}$); //将记录转成$|IA|$维数值空间的点
2. *CL*=*GDBSCAN*(Γ); //基于密度的空间聚类
3. *O*.*add*(*CL*.*popOutliers*(·)); //*popOutliers*(·)函数将 *CL* 中离群点取出, *add*(·)函数将离群点加入到离群点集合 *O* 中
4. **repeat**
5. *CL'*=Φ;
6. **for** *cl*=*CL*.*topCluster*(·) //取出最大类簇
7. *mbr*=*MBR*(*cl*); //将类簇生成 MBR
8. *cl'*=*MRTree*.*insert*(*mbr*); //见下面函数 *insert* 详情
9. **if** *cl'*!=Φ
10. *CL'*.*add*(*cl'*);

11. **if** $CL'!=\Phi$
12. $CL=GDBSCAN(CL')$;
13. $O.add(CL.popOutliers(\cdot))$; //同第 3 行
14. **until** $CL==\Phi$;
15. **for** $r_i \in O$ //将离群点数据逐一插入到 *MRTree*
16. $MRTree.insert(r_i)$;

函数. *insert*(*mbr*).

输入: 多属性索引 *MRTree*, 待插入的 MBR *mbr*;

输出: 剩余记录类簇 *cl*.

1. *LF*=*MRTree.ChooseLeaf*(*mbr.centre*); //*ChooseLeaf* 为 R-树中叶子节点搜索操作
2. 将 *mbr* 中符合 *LF* 中 MBR 约束的数据插入到 *LF*, 剩余数据集合为 *cl*;
3. 执行 *SplitNode*, 如果必要的话; //*SplitNode* 为 R-树中节点分裂操作, 当节点中数据量达到 *M* 时, 需要进行分裂操作
4. 执行 *AdjustTree*, 如果发生了节点分裂; //*AdjustTree* 为 R-树中向上传递操作, 调整树结构
5. 执行树增高操作, 如果 *AdjustTree* 导致根节点分裂;
6. **return** *cl*;

2.3 基于多属性索引的多维查询

整体而言, 多维查询将采用“过滤-验证”策略: 先利用多属性索引过滤掉不可能命中的记录, 得到候选记录集合; 然后采用相似性函数或距离函数逐一验证候选记录, 最后返回符合查询约束的记录. 多维查询将基于树的深度优先遍历进行, 利用递归实现. 给定$|IA|$维树形索引 *MRTree*, 其中, $|IA_{sim}|$维通过相似性函数衡量, $j \in IA_{sim}$, 第 j 个属性的相似性函数为 $sim^j(*,*)$, 相似性阈值为 θ^j; 剩余的$|IA_{dist}|=|IA|-|IA_{sim}|$维通过距离函数衡量, $k \in IA_{dist}$, 第 k 维的距离函数为 $dist^k(*,*)$, 距离阈值为 θ^k. 数值属性通常用距离函数来衡量, 它的距离下界可以直接计算得到. 如算法 3 所示, 给定查询记录 r_q, 那么$|IA|$维查询的具体流程如下.

(1) 叶子节点查询(第 2–4 行). 对当前叶子节点 T 中的每个记录 r_i 调用相似性函数或距离函数进行范围验证: ① 对于基于相似性衡量的$|IA_{sim}|$维中的每一维, 进行相似性约束验证(函数 *rangeVerify*, 第 1–3 行); ② 对于基于距离衡量的$|IA_{dist}|$维中的每一维, 进行距离约束验证(函数 *rangeVerify*, 第 4–6 行). 如果 $\wedge_{j \in IA_{sim}} sim^j(r_i^j, r_q^j) \geqslant \theta^j \wedge_{k \in IA_{dist}} dist^k(r_i^k, r_q^k) \leqslant \theta^k$ 为真, 将记录 r_i 加入查询结果; 否则, 过滤掉 r_i. 即: r_i 中任何一维不满足相似性或距离约束, 就过滤掉 r_i;

(2) 子树查询(第 6–8 行). 对当前非叶子节点 T 中的每个子节点 U 调用相似性上界函数或距离下界函数进行重叠验证: ① 对于基于相似性衡量的$|IA_{sim}|$维中的每一维, 进行相似性上界(记作 UB^j)约束验证(函数 *intersect*, 第 1–3 行); ② 对于基于距离衡量的$|IA_{dist}|$维中的每一维, 进行距离下界(记作 LB^k)约束验证(函数 *intersect*, 第 4–6 行), 其中, 距离下界可以直接计算得到, 将不对 LB^k 进行详述. 如果 $\wedge_{j \in IA_{sim}} UB^j \geqslant \theta^j \wedge_{k \in IA_{dist}} LB^k \leqslant \theta^k$ 为真, 那么继续对 U 指向的子节点进行查询; 否则, 过滤掉 U. 即: U 中任何一维不满足相似性或距离约束, 就过滤掉 U.

算法 3. 多维查询 *MultiDimenQuery*.

输入: 查询记录 r_q, 多属性索引节点 T, 索引属性编号集合 $IA=IA_{sim} \cup IA_{dist}$, 映射集合$\Xi=\{\xi_j | j \in IA\}$, 阈值集合$\Theta=\{\theta^j | j \in IA\}$;

输出: 查询结果 R_c.

1. **if** *T.type*==*LeafNode*
2. **for** $r_i \in T$
3. **if** $rangeVerify(r_q, r_i, IA, \Theta)$ //调用相似性或距离函数进行范围验证

4. R_c.*add*(r_i);
5. **else**
6. **for** $U \in T$.*childNodes*
7. **if** *intersect*(r_q,*MBR*(U),*IA*,Θ) //调用上(下)界函数进行重叠验证
8. *MultiDimenQuery*(r_q,U,*IA*,Ξ,Θ);

函数. *rangeVerify*(r_q,r,*IA*,Θ).

输入: 查询记录 r_q, 目标记录 r, 索引属性编号集合 $IA=IA_{sim} \cup IA_{dist}$, 阈值集合 $\Theta=\{\theta^j | j \in IA\}$;

输出: 范围验证结果.

1. **for** $j \in IA_{sim}$
2. **if** $sim^j(r^j, r_q^j) < \theta^j$ //调用相似性函数 sim^j 验证第 j 个属性
3. **return** false;
4. **for** $k \in IA_{dist}$
5. **if** $dist^k(r^k, r_q^k) > \theta^k$ //调用距离函数 $dist^k$ 验证第 k 个属性
6. **return** false;
7. **return** true;

函数. *intersect*(r_q,*mbr*,*IA*,Θ).

输入: 查询记录 r_q, 目标 MBR *mbr*, 索引属性编号集合 $IA=IA_{sim} \cup IA_{dist}$, 阈值集合 $\Theta=\{\theta^j | j \in IA\}$;

输出: 重叠验证结果.

1. **for** $j \in IA_{sim}$
2. **if** $UB^j(r_q^j, mbr_{\min}^j, mbr_{\max}^j) < \theta^j$ //调用相似性上界函数 UB^j 验证第 j 个属性
3. **return** false;
4. **for** $k \in IA_{dist}$
5. **if** $LB^k(r_q^k, mbr_{\min}^k, mbr_{\max}^k) > \theta^k$ //调用距离下界函数 LB^k 验证第 k 个属性
6. **return** false;
7. **return** true;

多维查询的调用方式为 *MultiDimenQuery*(r_q,*MRTree.root*,*IA*,Ξ,Θ), 从根节点 *MRTree.root* 开始, 找出所有符合查询条件的记录. 通过分析算法 3 可知, 相似性上界计算的有效性和效率将决定索引的过滤能力. 为此, 需要设计高效的"字符串→数值"映射.

3 "字符串→数值"映射

为了将字符串插入到树形索引, 需要将字符串映射到数值空间, 将映射记作 ξ. 根据命题 1, ξ必须满足两点: ① 字符串集合中任意两个字符串映射后存在大小关系, 即可比较性, 这是构建多属性树形索引的基础条件; ② 给定字符串相似性函数 *sim*(*,*), 可以快速地计算出给定字符串 t_q 与一个字符串范围 $rng=[t_i,t_j]_\xi$的最大相似性, 即相似性上界可计算, 这将决定索引的性能. 第 3.1 节提出了一个基本的"字符串→数值"映射方法; 第 3.2 节提出了两个降维策略来优化基本的映射方法; 在第 3.2 节的基础上, 第 3.3 节通过 z-order 变换作进一步优化, 得到最终的映射方法.

3.1 基于q-gram的映射

字符串相似性分为两类: 基于编辑的相似性(如编辑相似性)和基于集合的相似性(如 Jaccard 相似性). 这两类字符串相似性都与 q-gram 有着密切的关系[21−23]: 前者可以通过 q-gram 来估计, 后者则需要通过 q-gram 来计算. 因此, 借助 q-gram 实现"字符串→数值"映射. 将以编辑相似性和 Jaccard 相似性为代表, 给出基于 q-gram 的映射方法. 同类其他的相似性将作为未来工作.

如例 2 所示, Ω为全局 q-gram 的集合(即字符串集合中所有 q-gram 的集合), 给定字符串 t_q 的 q-gram 级标识集合为 $G(t_q)$, 用一个$|\Omega|$维的标识向量 $\boldsymbol{v}_{tq}$ 统计每个 q-gram 的出现频率; 然后, 将向量 $\boldsymbol{v}_{tq}$ 每个维度换算成二进制, 拼接成一个更长的二进制数字(例 2 中 10001000100000110001001101), 换算成十进制后通常是一个巨大的数字(例 2 中 35785805). 这个基于 q-gram 的映射方法显然符合可比较性(引理 1), 并且对于编辑相似性和 Jaccard 相似性来说, 上界可计算(引理 2 和引理 3). 然而, 由于标识向量具有超高维特点, 这种映射方法不仅存储代价过高, 而且相似性上界计算开销过高, 具有显著的可优化空间.

例 2: Ω={g_0,g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8,g_9,g_{10},g_{11},g_{12},g_{13},g_{14},g_{15},g_{16},g_{17},g_{18},g_{19},g_{20},g_{21},g_{22},g_{23}}, $G(t_q)$={g_0,g_4,g_4,g_7,g_{13},g_{13},g_{13},g_{17},g_{20},g_{21},g_{23}}, $\boldsymbol{v}_{tq}$=(1,0,0,0,2,0,0,1,0,0,0,0,0,0,3,0,0,0,1,0,0,1,1,0,1).

$\boldsymbol{v}_{tq}$ 对应的二进制: 10001000100000110001001101, $\boldsymbol{v}_{tq}$ 对应的十进制: 35785805.

对于基于 q-gram 的映射方法, 编辑相似性上界计算复杂度为 O($|\Omega|$), Jaccard 相似上界的计算复杂度为 O($|\Omega|$).

引理 1. 基于 q-gram 的映射满足可比较性(证明略).

引理 2. 基于 q-gram 的映射满足编辑相似性上界可计算.

证明: 字符串的编辑距离与其 q-gram 集合存在很强的联系. 给定两个字符串 t_i 和 t_j, q-gram 集合分别为 $G(t_i)$和 $G(t_j)$, 那么两者的编辑距离满足以下关系[21−23]:

$$ED(t_i,t_j)\geqslant \max\left(\frac{|G(t_i)-G(t_j)|}{q},\frac{|G(t_j)-G(t_i)|}{q}\right) \tag{4}$$

其中, $G(t_i)$–$G(t_j)$表示出现在 $G(t_i)$中而没有出现在 $G(t_j)$中的 q-gram 组成的集合. 那么, 编辑相似性将满足:

$$ES(t_i,t_j)=1-\frac{ED(t_i,t_j)}{\max(|t_i|,|t_j|)}\leqslant 1-\frac{\max(|G(t_i)-G(t_j)|/q,|G(t_j)-G(t_i)|/q)}{\max(|t_i|,|t_j|)} \tag{5}$$

其中, 编辑相似性上界的计算复杂度为 O($|\Omega|$). □

引理 3. 基于 q-gram 的映射满足 Jaccard 相似性上界可计算.

证明: Jaccard 相似性本身就是根据 q-gram 集合计算而来的, 因此满足上界可计算. 其中, Jaccard 相似性上界的计算复杂度为 O($|\Omega|$). □

3.2 基于降维的映射优化

为了降低存储开销和相似性上界计算开销, 可以将超高维标识向量转换为低维向量. 有两种降维思路: 基于哈希的降维和基于频率分类的降维[4−6].

- 基于哈希的降维. 如例 3(1)所示, 利用一个哈希函数将全局 q-gram 映射到 L_{hs} 个哈希桶中(L_{hs}<<$|\Omega|$). 给定字符串 t_q, 统计 t_q 的 q-gram 在每个桶中出现的总次数, 组成一个 L_{hs} 维哈希向量 v_{hs} 来表示字符串 t_q. 在基于哈希的降维中, 向量 v_{hs} 的各个维度是随机、平等的, 从概率上保证每个维度中 q-gram 的出现频率相差不大;
- 基于频率分类的降维. 如例 3(2)所示, 将全局 q-gram 按出现频率升序排列, 然后划分成 L_{fc} 类(L_{fc}<<$|\Omega|$), 每一类内 q-gram 的出现频率相近, 每一类都装入一个桶内, 将桶按其中 q-gram 的出现频率升序排列. 给定字符串 t_q, 统计 t_q 的 q-gram 在每个桶中出现的总次数, 组成一个 L_{fc} 维频率向量 v_{fc} 来表示字符串 t_q. 在基于频率分类的降维中, 向量 v_{fc} 的维度是有序的, 其逻辑为: 若一个 q-gram 的出现频率越高, 那么它对于字符串过滤或相似性计算的作用越小, 因此应该优先考虑出现频率低的 q-gram.

例 3: Ω={g_0,g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8,g_9,g_{10},g_{11},g_{12},g_{13},g_{14},g_{15},g_{16},g_{17},g_{18},g_{19},g_{20},g_{21},g_{22},g_{23}}, $G(t_q)$={g_0,g_4,g_4,g_7,g_{13},g_{13},g_{13},g_{17},g_{20},g_{21},g_{23}}, 那么两种降维方法的计算过程及结果如下.

(1) 基于哈希的降维, L_{hs}=3.

- 全局哈希: B_{hs0}={g_0,g_2,g_5,g_{10},g_{13},g_{15},g_{19},g_{23}}, B_{hs1}={g_1,g_3,g_4,g_6,g_9,g_{14},g_{18},g_{21}}, B_{hs2}={g_7,g_8,g_{11},g_{12},g_{16},g_{17},g_{20},g_{22}};
- t_q 的哈希向量: $\boldsymbol{v}_{hs}$=(5,3,3);

(2) 基于频率分类的降维, L_{fc}=3.

- 基于频率升序的全局 q-gram 排序: g_0, g_7, g_{21}, g_3, g_9, g_{12}, g_{16}, g_{19}, g_{20}, g_6, g_{11}, g_{14}, g_{18}, g_{22}, g_{23}, g_2, g_8, g_{15}, g_{17}, g_1, g_4, g_5, g_{13}, g_{10};
- 等频率小组: f_0={g_0,g_7,g_{21},g_3,g_9,g_{12},g_{16},g_{19}}, f_1={g_{20},g_6,g_{11},g_{14},g_{18},g_{22}}, f_2={g_{23},g_2,g_8,g_{15}}, f_3={g_{17},g_1,g_4}, f_4={g_5,g_{13},g_{10}}. 其中, 每个小组 f 中的 q-gram 的频率相等, 频率大小关系: f_0<f_1<f_2<f_3<f_4;
- 按频率分类并装桶: B_{fc0}={g_0,g_7,g_{21},g_3,g_9,g_{12},g_{16},g_{19},g_{20},g_6,g_{11}}, B_{fc1}={g_{14},g_{18},g_{22},g_{23},g_2,g_8,g_{15},g_{17}}, B_{fc2}={g_1,g_4,g_5,g_{13},g_{10}};
- t_q 的频率向量: $\boldsymbol{v}_{\boldsymbol{fc}}$=(4,2,5).

基于哈希的映射和基于频率分类的映射显然都满足可比较性; 它们分别满足编辑相似性上界可计算和 Jaccard 相似上界可计算, 具体参考引理 4–引理 7. 本质上, 基于降维的优化是用索引过滤的有效性(相似性上界的精确度)来换取过滤开销, 作为可调节参数, L_{hs} 或 L_{fc} 越大, 则过滤能力越强, 而过滤开销也越大. L_{hs} 与 L_{fc} 的最佳取值可以通过实验测定. 对于基于哈希的映射方法, 编辑相似性上界的计算复杂度为 O(L_{hs}), Jaccard 相似上界的计算复杂度为 O(L_{hs}). 对于基于频率分类的映射方法, 编辑相似性上界的计算复杂度为 O(L_{fc}), Jaccard 相似上界的计算复杂度为 O(L_{fc}).

引理 4. 基于哈希的映射满足编辑相似性上界可计算.

证明: 给定两个字符串 t_i 和 t_j, 将 q-gram 按哈希桶分别进行统计, 根据引理 2 中的公式(4), 那么两者的编辑距离满足:

$$ED(t_i,t_j)\geqslant \max_{l\in[0,L_{hs}-1]}\left(\sum_{v^i_{hs}[l]\geqslant v^j_{hs}[l]}\frac{v^i_{hs}[l]-v^j_{hs}[l]}{q},\sum_{v^j_{hs}[l]\geqslant v^i_{hs}[l]}\frac{v^j_{hs}[l]-v^i_{hs}[l]}{q}\right) \tag{6}$$

那么, 编辑相似性满足:

$$ES(t_i,t_j)\leqslant 1-\frac{\max_{l\in[0,L_{hs}-1]}\left(\sum_{v^i_{hs}[l]\geqslant v^j_{hs}[l]}\frac{v^i_{hs}[l]-v^j_{hs}[l]}{q},\sum_{v^j_{hs}[l]\geqslant v^i_{hs}[l]}\frac{v^j_{hs}[l]-v^i_{hs}[l]}{q}\right)}{\max(|t_i|,|t_j|)} \tag{7}$$

其中, 编辑相似性上界的计算复杂度为 O(L_{hs}). □

引理 5. 基于哈希的映射满足 Jaccard 相似性上界可计算.

证明: 根据 Jaccard 相似性定义及基于哈希的降维原理, Jaccard 相似性满足:

$$Jaccard(t_i,t_j)\leqslant 1-\frac{\sum_{l\in[0,L_{fc}-1]}|v^i_{fc}[l]-v^j_{fc}[l]|}{|t_i|+|t_j|} \tag{8}$$

其中, Jaccard 相似性上界的计算复杂度为 O(L_{hs}). □

引理 6. 基于频率分类的映射满足编辑相似性上界可计算. 证明略.

引理 7. 基于频率分类的映射满足 Jaccard 相似性上界可计算. 证明略.

引理 6 与引理 4 的证明类似, 引理 7 与引理 5 的证明类似, 不再赘述.

- 动态数据环境讨论.

基于哈希的降维, 与 q-gram 的频率无关, 因此支持动态数据环境. 基于频率分类的降维, 与 q-gram 的频率相关, 而数据变化带来 q-gram 的频率变化, 因此该方法不支持动态数据环境. 本文主要关注静态数据环境, 但如何改进基于频率分类的降维以适应动态数据环境, 是值得探索的问题. 一个基本方法是: 当数据变化后, 重新统计 q-gram 频率, 根据频率调整排序与分组, 然而这种方式开销较大. 为此, 可以采用一种基于分类偏离的解决思路. 如下面公式(9)所示, 定义一个分类偏离值 τ, 衡量数据更新前后频率分类的总体差异性. 其中, $BNo^{now}(g_i)$表示数据更新前 g_i 所属分类编号, $BNo^{new}(g_i)$表示数据更新后 g_i 所属分类编号, ΔG 是数据更新导致所属分类发生变化的 q-gram 集合. 分类偏离值 τ可以用于决定是否需要进行频率分类调整:

$$\tau=\sum_{g_i\in\Delta G}|BNo^{new}(g_i)-BNo^{now}(g_i)| \tag{9}$$

3.3 基于*z*-order的映射优化

基于降维向量的映射还有进一步的优化空间. 给定两个正整数, 将它们转换为二进制进行比较, 需要从高位到低位依次比较, 如果高位出现不相等的情况, 则剩余位数不必再作比较[4]. 基于这个启发, 利用 *z*-order 优化相似性上界计算.

例 4: 如图 3(a)所示, 字符串 t_q 的表示向量 $\boldsymbol{v_q}$=(5,3,3), 将向量 $\boldsymbol{v_q}$ 每一维都转换成二进制, 然后执行 *z*-order 变换, 得到一个新的 0-1 序列 100011111. 现在有一个一维查询: 查询串为 t_q, 相似性函数为 Jaccard, 阈值为 0.8. 给定字符串范围$[t_i,t_j]$的 *z*-order 序列表示是[010101011,010101101], 可以表示为 010101???, 其中, 010101 为共同前缀, ?可能取 0 或 1, 那么如图 3(b)所示. 010101???中任意字符串 t_{ij} 对应向量 $\boldsymbol{v_{ij}}$, $\boldsymbol{v_{ij}}$[0]可能取 2 或 3, $\boldsymbol{v_{ij}}$[1]可能取 4 或 5, $\boldsymbol{v_{ij}}$[2]可能取 2 或 3. 现在计算 t_q 与$[t_i,t_j]$的 Jaccard 相似性上界, 上界向量不妨记作 $\boldsymbol{v_{max}}$. $\boldsymbol{v_q}$[0]=5>3, 那么 $\boldsymbol{v_{max}}$[0]=3; $\boldsymbol{v_q}$[1]=3<4, 那么 $\boldsymbol{v_{max}}$[1]=4; $\boldsymbol{v_q}$[2]=3, 那么 $\boldsymbol{v_{max}}$[2]=3; 因此 $\boldsymbol{v_{max}}$=(3,4,3), t_q 与$[t_i,t_j]$的相似性上界为 9/(11+10−9)=0.75<0.8, 所以字符串范围$[t_i,t_j]$内不可能存在符合查询条件的字符串, 可以直接过滤掉. 同理, 编辑相似性的上界也可以算出, 这里不再赘述.

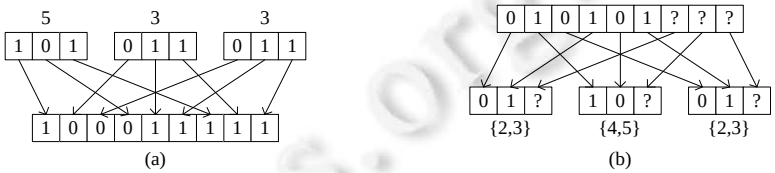


图 3 基于 *z*-order 优化的示例

给定相似性 *sim*, 基于 *z*-order 的相似性上界计算如算法 4 所示. 首先计算基于 *z*-order 映射的二进制表示(第 1 行); 计算字符串范围上、下界的共同前缀长度 *pre*(第 2 行); 将字符串范围下界后 *L-pre* 位设置为 0, 字符串范围上界后 *L-pre* 位设置为 1(第 3–5 行); 利用公式(7)或公式(8)计算 $b_q^{\xi_z}$ 与 $[b_{\min}^{\xi_z},b_{\max}^{\xi_z}]$ 的 *sim* 相似性上界 *ub*, 作为 t_q 与 $[t_{\min},t_{\max}]_{\xi_z}$ 的 *sim* 相似性上界(第 6 行).

以基于低维向量的映射为基础, 基于 *z*-order 的映射从技术实现的角度, 进一步提升了过滤能力. 虽然理论上, 基于 *z*-order 的映射与基于低维向量的映射的相似性上界计算复杂度是相同的, 但在实际运行过程中, 基于 *z*-order 映射的相似性上界计算开销要更小.

算法 4. 基于 *z*-order 映射ξ_z的 *sim* 相似性上界计算.

输入: 查询字符串 t_q, 字符串范围 $[t_{\min},t_{\max}]_{\xi_z}$;

输出: t_q 与 $[t_{\min},t_{\max}]_{\xi_z}$ 的 *sim* 相似性上界.

1. 计算 t_q, $t_{\min}$, $t_{\max}$ 基于映射ξ_z的二进制表示 $b_q^{\xi_z}$, $b_{\min}^{\xi_z}$, $b_{\max}^{\xi_z}$;
2. 计算 $b_{\min}^{\xi_z}$ 与 $b_{\max}^{\xi_z}$ 的共同前缀长度 *pre*;
3. **for** *j*=*pre* to *L*−1
4. $b_{\min}^{\xi_z}[j]=0$;
5. $b_{\max}^{\xi_z}[j]=1$;
6. 计算 $b_q^{\xi_z}$ 与 $[b_{\min}^{\xi_z},b_{\max}^{\xi_z}]$ 的 *sim* 相似性上界 *ub*; //Jaccard 相似性采用公式(8), 编辑相似性则采用公式(7).
7. **return** *ub*;

4 实验评价

4.1 实验设置

实验代码通过 C++实现; 编译环境: GCC 4.8; 运行环境: 处理器 3.2 GHz Intel (R) E5-2667, 8 核, 内存 64 GB; 操作系统: Red Hat Linux.

- 数据集.

实验评价使用两个数据集: (1) 引文数据集 DBLP, 从 DBLP (https://dblp.uni-trier.de/)数据库抓取引文记录, 选中某些记录以概率方式生成重复记录, 最终得到 200k 条引文记录, 其中包含 136 941 对重复记录, 通过标题、作者、期刊/会议、年份这 4 个属性描述; (2) 个人信息数据集 FB, 利用 Febrl 数据生成器构建一个合成的个人信息数据集[24], 包含 1 000k 条个人记录, 其中包含 974 360 对重复记录, 通过姓名、性别、出生年、住址、城市、州和邮编这 8 个属性描述. 重复记录生成方法: 将原记录复制, 通过对复制记录中选定属性值进行字符级增加、删除、替换等操作生成重复记录.

- 索引设置.

对于 DBLP 数据, 构建三维索引: 标题基于 Jaccard 相似性, 第一作者基于编辑相似性, 年份基于差值. 对于 FB 数据, 构建三维索引: 姓名基于编辑相似性, 州+城市基于 Jaccard 相似性, 出生年基于差值. 降维优化中, 通过实验测定 L_{hs}=L_{fc}=10. 设定 6 个多维查询的阈值组合, 见表 1, 从 rg_1 到 rg_6, 逐渐放松阈值约束.

表 1 多维查询阈值

	rg_1	rg_2	rg_3	rg_4	rg_5	rg_6
Jaccard	0.8	0.7	0.7	0.7	0.6	0.6
ES	0.8	0.8	0.7	0.7	0.7	0.6
差值	1	1	1	2	2	2

4.2 分项测试

针对本文提出的多属性索引技术的多个方面进行测试, 包括参数测试、索引构建、映射方法、伸缩性和存储开销等方面.

(1) 参数测试.

降维优化中, 参数 L_{hs} 与 L_{fc} 通过实验测定. 为了便于阅读, 图 4–图 7 只展示了多维查询中阈值取 rg_1、rg_3 和 rg_5 的实验结果, 但实际进行了 rg_1 到 rg_6 所有阈值的实验, 整体的实验结果变化趋势与当前展示的实验结果是一致的.

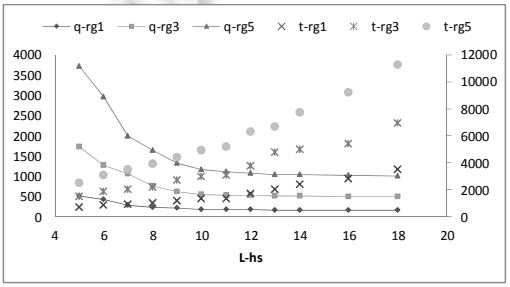


图 4 DBLP 数据集上参数 L_{hs} 测试

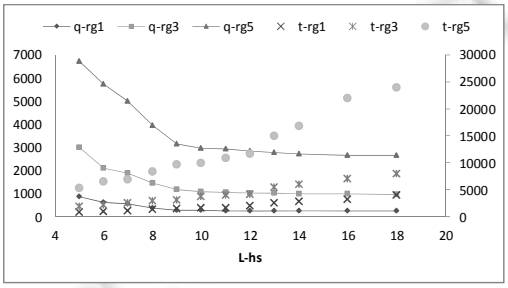


图 5 FB 数据集上参数 L_{hs} 测试

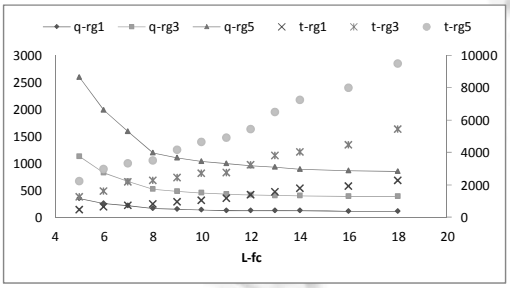


图 6 DBLP 数据集上参数 L_{fc} 测试

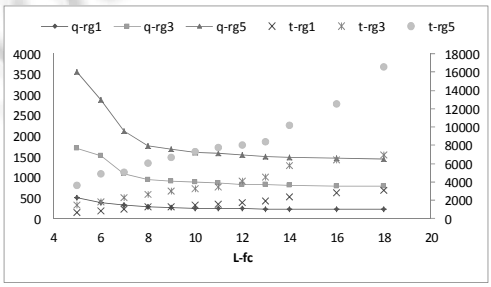


图 7 FB 数据集上参数 L_{fc} 测试

在 DBLP 数据上, 进行 5k 次随机抽样, 并利用抽样记录进行 5k 次多维查询; 在 FB 数据上, 进行 25k 次随机抽样, 并利用抽样记录进行 25k 次多维查询. 分别对多维查询的候选记录集合规模和时间开销取均值作为测试结果. 图 4–图 7 中: 候选记录集合规模通过细线连接的散点表示, 对应主纵坐标; 时间开销通过散点表示, 对应从纵坐标. 图 4 和图 5 分别表示取不同的 L_{hs} 维度时, MRT-hs-z 在 DBLP 数据和 FB 数据上的表现. 图 4 中, 当 L_{hs} 增加到 10 至 11 时, 候选记录集合规模变化趋缓, 而时间开销一直不断增长; 图 5 中, 当 L_{hs} 增加到 9 至 10 时, 候选记录集合规模变化趋缓, 而时间开销一直不断增长. 图 6 和图 7 分别表示取不同的 L_{fc} 维度时, MRT-fc-z 在 DBLP 数据和 FB 数据上的表现. 图 6 中, 当 L_{fc} 增加到 9 至 10 时, 候选记录集合规模变化趋缓, 而时间开销一直不断增长; 图 7 中, 当 L_{fc} 增加到 8 至 9 时, 候选记录集合规模变化趋缓, 而时间开销一直不断增长. 总体而言, L_{hs} 或 L_{fc} 取到 8 至 12 范围内, 相应的候选记录集合规模降到了比较理想的取值范围, 此时, 如果 L_{hs} 或 L_{fc} 维度继续增长, 则候选记录集合规模变化很小, 但时间开销会持续增长, 甚至可能会增长变快. 比如图 4 中, 当 L_{hs} 增加到 12 至 13 时, 时间开销增长变快. 综上分析, 本文令 L_{hs}=L_{fc}=10 是一个合理的设置, 取得了候选记录集合规模与时间开销的良好平衡.

(2) 索引构建.

在查询式实体解析中需要在线构建索引, 为此, 将索引构建场景设定如下: 缓存记录以小批量(每次 500 条)的形式产生, n 个索引构建方法, 当其中最快的构建方法完成本批记录插入时, 触发新一批缓存记录的到来. 本文提出的基于空间聚类的在线 R-树构建方法记作 cstr-online, 传统的动态 R-树构建方法记作 cstr-tr[9], 还有两个动态 R-树构建方法: Li 等人提出的 LBI 方法[18]和 Lee 等人提出的 SCB 方法[19]. 图 8 和图 9 分别呈现了在 DBLP 数据集和 FB 数据集上 4 种索引构建方法的表现. 整体来看, 在两个数据集上, cstr-online 的实时时间开销都明显小于 cstr-tr、LBI 和 SCB. 累计来看: 在 DBLP 数据集上, cstr-online 的时间开销比 cstr-tr 小 45%、比 LBI 小 33%、比 SCB 小 20%; 在 FB 数据集上, cstr-online 的时间开销比 cstr-tr 小 73%、比 LBI 小 57%、比 SCB 小 34%. 综上, 相比于 cstr-tr、LBI 和 SCB, cstr-online 更能满足查询式实体解析中在线构建索引的需求.

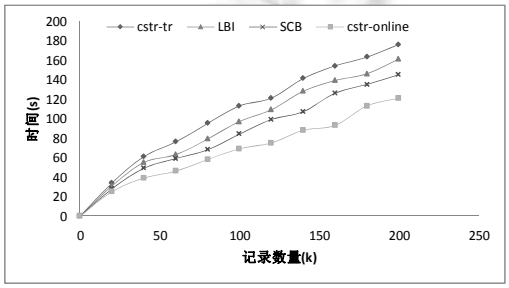


图 8 DBLP 数据集上小批量式在线构建测试

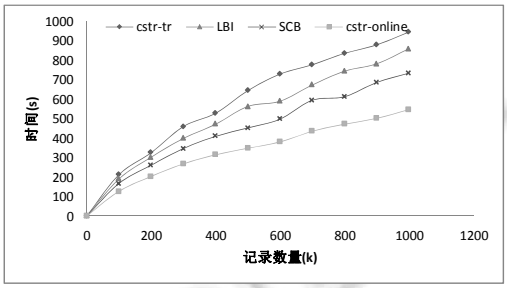


图 9 FB 数据集上小批量式在线构建测试

(3) “字符串→数值”映射.

本小节比较不同的映射方法: MRT-hs 采用基于哈希的降维; MRT-hs-z 采用基于哈希的降维, 并进行 z-order 优化; MRT-fc 采用基于频率分类的降维; MRT-fc-z 采用基于频率分类的降维, 并进行 z-order 优化. 在 DBLP 数据集上进行 5k 次随机抽样, 并利用抽样记录进行 5k 次多维查询, 每次多维查询的候选记录数量取均值后得到的结果如图 10 所示; 在 FB 数据集上进行 25k 次随机抽样, 并利用抽样记录进行 25k 次多维查询, 每次多维查询的候选记录数量取均值后得到的结果如图 11 所示. 对于每一种索引技术, 从 rg_1 到 rg_6, 阈值逐渐变小, 从而产生更多的候选记录. 阈值相同的情况下, 比较 4 种索引技术产生的候选记录数量. 整体而言, 4 种索引技术的过滤能力如下: MRT-hs<MRT-fc<MRT-hs-z<MRT-fc-z. 说明: ① 在过滤能力方面, 基于频率分类降维的优化比基于哈希降维的优化更有效; ② z-order 优化有助于提升索引的过滤能力.

(4) 可伸缩性.

本小节测试多属性索引 MRT-fc-z 的可伸缩性(scalability). 对于 DBLP 数据, 在 40k、80k、120k、160k 和 200k 的数据规模上, 分别进行 1k、2k、3k、4k 和 5k 次随机抽样, 并利用抽样记录分别进行 1k、2k、3k、4k

和 5k 次多维查询, 每次多维查询的时间开销取均值后得到的结果如图 12 所示; 对于 FB 数据, 在 200k、400k、600k、800k 和 1 000k 的数据规模上, 分别进行 5k、10k、15k、20k 和 25k 次随机抽样, 并利用抽样记录分别进行 5k、10k、15k、20k 和 25k 次多维查询, 每次多维查询的时间开销取均值后得到的结果如图 13 所示. 整体而言: 在两个数据集上, MRT-fc-z 在不同阈值设定下, 随着数据量的增长, 取得近似线性的时间开销增长, 因此具有良好的可伸缩性.

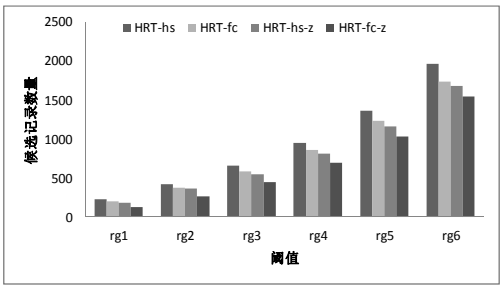


图 10 DBLP 数据集上映射方法对比

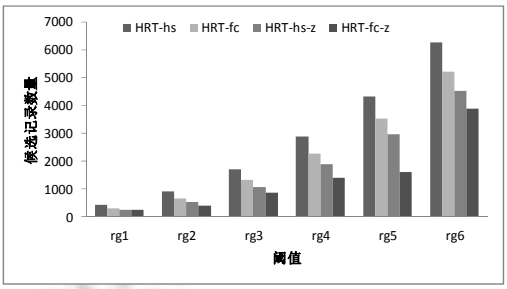


图 11 FB 数据集上映射方法对比

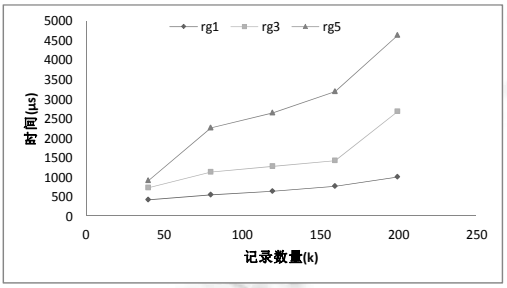


图 12 DBLP 数据集上可伸缩性测试

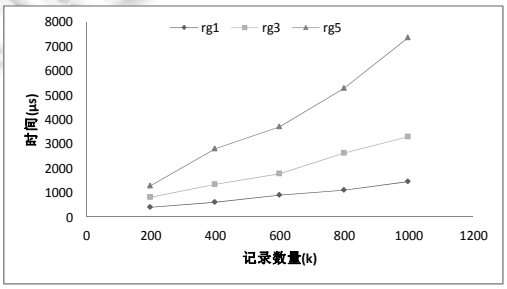


图 13 FB 数据集上可伸缩性测试

(5) 存储开销.

本小节测试多属性索引 MRT-fc-z 的存储开销. 对于 DBLP 数据, 在 40k、80k、120k、160k 和 200k 的数据规模上, 分别进行索引构建并测量其存储开销, 得到的结果如图 14 所示, 其增长趋势近似于线性; 对于 FB 数据, 在 200k、400k、600k、800k 和 1 000k 的数据规模上, 分别进行索引构建并测量其存储开销, 得到的结果如图 15 所示, 其增长趋势也近似于线性. 综上分析, 在两个数据集上, 随着数据量的增长, MRT-fc-z 的存储开销取得近似线性增长, 因此具备良好的存储可伸缩性.

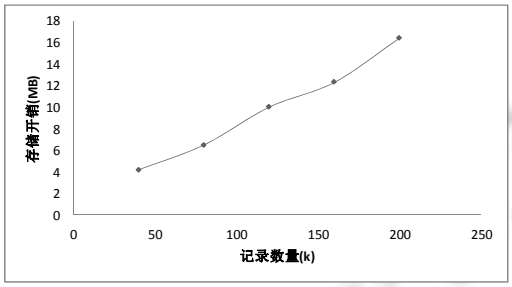


图 14 DBLP 数据集上存储开销测试

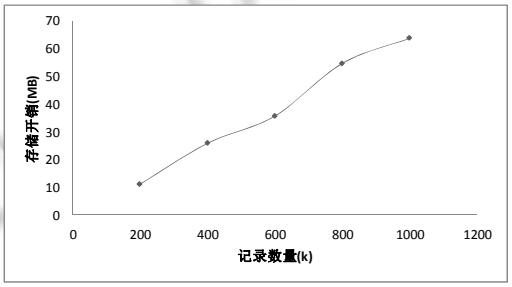


图 15 FB 数据集上存储开销测试

4.3 与已有工作对比

将本文提出的多属性索引技术记作 MRT-fc-z, 采用基于频率分类的降维, 并进行 *z*-order 变换. 将 MRT-fc-z 与 Li 等人提出的基于前缀树的索引技术 pre-T[7]、Ardalan 等人提出的 smurf 方法[25]进行比较. 对于 DBLP

数据中的年份和 FB 数据中的出生年, pre-T 和 smurf 采用编辑距离衡量[7,25]. 在 DBLP 数据集上进行 5k 次随机抽样, 并利用抽样记录进行 5k 次多维查询, 每次多维查询的候选记录数量取均值后得到的结果如图 16 所示; 在 FB 数据集上进行 25k 次随机抽样, 并利用抽样记录进行 25k 次多维查询, 每次多维查询的候选记录数量取均值后得到的结果如图 17 所示. 在 DBLP 数据集上, 从 rg_1 到 rg_6, MRT-fc-z 的候选集要明显小于 pre-T 和 smurf, 比例分别至少为 30%和 17%. 同样, 在 FB 数据集上, 从 rg_1 到 rg_6, MRT-fc-z 的候选集要明显小于 pre-T 和 smurf, 比例分别至少为 34%和 23%. 综合来看, 相比于 pre-T 和 smurf, MRT-fc-z 在过滤能力方面具有显著的优势.

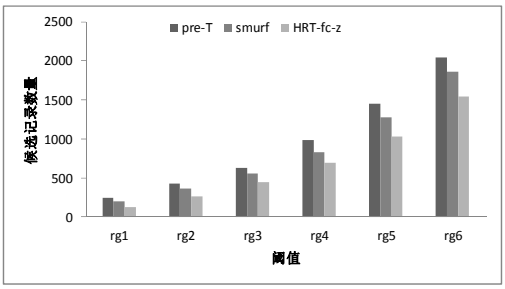


图 16 DBLP 数据集上与已有工作对比

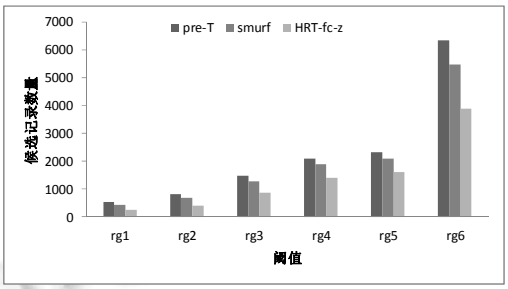


图 17 FB 数据集上与已有工作对比

5 相关工作

实体解析也称为实体匹配、记录链接等, 是数据集成和数据预处理的一个重要方面[1,2]. 随着大数据产生速度和更新频率的不断提升, 查询式实体解析成为实体解析中的热点问题. Christen 等人提出了基于相似性索引的查询式实体解析方法 SAII, 其核心思想是: 提前计算同一块内的属性相似性, 并将属性相似性存储在内存, 以便查询-解析过程中直接使用[26]. Ramadan 等人提出了基于动态滑动窗口索引的查询式实体解析方法 SNMI, 基于 AVL 树的索引按字母顺序存储块, 在查询-解析过程中将查询记录插入到该索引结构中, 并利用窗口算法来产生候选记录[3]. SNMI 包含多种窗口算法: 固定窗口、基于相似性的自适应窗口、基于候选成员的自适应窗口等. Altwaijry 等人将实体解析与数据库中 SQL 查询的 Select-Project-Join (SPJ)操作结合起来, 提出了共同解决方案[27]. Dragut 等人对 Web 查询的结果实时地进行实体解析和数据融合, 利用缓存来减少计算开销[28].

索引是查询处理中常用的数据过滤工具. 字符串相似性搜索中常用的索引有倒排索引、基于 B 树的索引、字典树等, 大部分都是一维索引[4], 然而查询式实体解析中需要对缓存记录进行多属性数据索引. Li 等人提出了针对多属性数据的连接和查询方法 pre-T, 其核心为基于前缀树的索引[7]. pre-T 通过一个代价模型来引导构建高质量的前缀树. 对于多属性查询, 需要构造多个前缀树索引. pre-T 可以针对数据记录中的不同属性采用不同的字符串相似性, 比如 Jaccard、Overlap、编辑距离等, 但不支持数值比较. Jagadish 等人提出了支持通配符的多维字符串索引, 但不支持相似性查询[29]. Zhang 等人提出了基于编辑距离的字符串索引 B^{ed}-Tree(一维), 将字符串实现某种排序后, 按照排序结果将字符串插入 B+树, 从而实现字符串查询过程中的数据过滤[5]. B^{ed}-Tree 设计了 3 种字符串排序: 基于字典顺序的排序、基于 gram 计数的排序和基于 gram 位置的排序. Zhang 等人提出了基于集合相似性的字符串索引(一维), 将字符串转换为数值插入到 B+树中, 并提出动态的数据划分算法以提升过滤能力[6].

基于树的空间索引(如 R-树、KD-树和四分树等)支持多维索引[8], 但只适用于数值, 不适用于字符串. 作为多维索引的代表, R-树将 B-树的思想拓展到多维空间, 有效地解决了多维空间的搜索问题, 在数据库、地理信息系统等领域广受欢迎. R-树支持快速的多维联合查询, 支持高效的插入、删除等更新操作[9]. 它是平衡树, 在插入、删除操作时采用合并、分裂节点的方法, 保证树的平衡性. R-树采用最小临界矩形(MBR)方法对多维空间进行层次分割: 从叶子节点开始用 MBR 将空间框起来, 节点越往上, 框住的空间就越大. 它的非叶子节

点不存储数据，只有叶子节点指向数据，可以很好地支持外存算法.

传统的实体解析依赖于经典的字符串相似性方法和数值相似性方法[1]，本文提出的多属性数据索引技术主要面向这类语法级相似性. 然而，当前出现了基于语义相似性的实体解析方法，如基于深度学习的实体解析[30]，这类方法的解析质量更高. 本文的索引技术无法应对此类相似性，如何构建面向语义级相似性的多属性数据索引，是未来非常有价值的研究方向.

6 总 结

针对查询式实体解析中的缓存查询需求，本文提出了基于 R-树的多属性数据索引技术 MRTree. 设计了树形的多属性索引结构，通过“字符串→数值”映射将字符串属性插入索引，在此基础上定义多维查询. 针对 Jaccard 相似性和编辑相似性，提出了一组“字符串→数值”映射及优化方法，保证多属性索引的过滤能力. 未来的工作将从以下几个方面展开.

1) 将研究适用于其他字符串相似性的“字符串→数值”映射方法，并将探索实体缓存调度问题;
2) 将研究面向语义相似性的多属性数据索引技术;
3) 将对“字符串→数值”映射中的相关方法加以进一步探索: 将基于频率分类的降维方法推广到动态数据上; 针对降维后的向量的可比较性，探索更准确的衡量指标; 研究理论化的 L_{hs} 与 L_{fc} 的最佳取值设置方法.

References:

[1] Elmagarmid AK, Ipeirotis PG, Verykios VS. Duplicate record detection: A survey. IEEE Trans. on Knowledge and Data Engineering, 2007, 19(1): 1−16. [doi: 10.1109/TKDE.2007.250581]

[2] Sun CC, Shen DR, Li YK, Xiao YY, Ma JH. Research on record pair ranking for entity resolution with time constraint. Ruan Jian Xue Bao/Journal of Software, 2020, 31(3): 695−709 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5900.htm [doi: 10.13328/j.cnki.jos.005900]

[3] Ramadan B, Christen P, Liang H, Gayler RW. Dynamic sorted neighborhood indexing for real-time entity resolution. Journal of Data and Information Quality (JDIQ), 2015, 6(4): 15.

[4] Yu M, Li G, Deng D, Feng J. String similarity search and join: A survey. Frontiers of Computer Science, 2016, 10(3): 399−417.

[5] Zhang Z, Hadjieleftheriou M, Ooi BC, Srivastava D. Bed-tree: An all-purpose index structure for string similarity search based on edit distance. In: Proc. of the 2010 ACM SIGMOD Int’l Conf. on Management of Data. 2010. 915−926.

[6] Zhang Y, Li X, Wang J, Zhang Y, Xing C, Yuan X. An efficient framework for exact set similarity search using tree structure indexes. In: Proc. of the 33rd IEEE Int’l Conf. on Data Engineering (ICDE). 2017. 759−770.

[7] Li G, He J, Deng D, Li J. Efficient similarity join and search on multi-attribute data. In: Proc. of the 2015 ACM SIGMOD Int’l Conf. on Management of Data. 2015. 1137−1151.

[8] Gaede V, Günther O. Multidimensional access methods. ACM Computing Surveys (CSUR), 1998, 30(2): 170−231.

[9] Guttman A. R-trees: A dynamic index structure for spatial searching. In: Proc. of the 1984 ACM SIGMOD Int’l Conf. on Management of Data. 1984. 47−57.

[10] Shao J, Wang Q, Lin Y. Skyblocking for entity resolution. Information Systems, 2019, 85: 30−43.

[11] Christen P. A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. on Knowledge and Data Engineering, 2012, 24(9): 1537−1555.

[12] Hassanzadeh O, Chiang F, Lee HC, Miller RJ. Framework for evaluating clustering algorithms in duplicate detection. Proc. of the VLDB Endowment, 2009, 2(1): 1282−1293.

[13] Li Y, Gao J, Meng C, Li Q, Su L, Zhao B, Fan W, Han J. A survey on truth discovery. SIGKDD Explorations, 2015, 17(2): 1−16.

[14] Tian H, Sheng W, Shen H, Wang C. Truth finding by reliability estimation on inconsistent entities for heterogeneous data sets. Knowledge-based Systems, 2020, 187: 104828.

[15] Sun CC, Shen DR, Kou Y, Nie TZ, Yu G. A related data oriented joint entity resolution approach. Chinese Journal of Computers, 2015, 38(9): 1739−1754 (in Chinese with English abstract).

[16] Kamel I, Faloutsos C. Hilbert R-tree: An improved R-tree using fractals. In: Proc. of the 20th VLDB. 1994. 500−509.

[17] Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries. In: Proc. of the ACM SIGMOD. 1995. 71−79.

[18] Li C, Rupesh C, Elke AR. Merging R-trees: Efficient strategies for local bulk insertion. GeoInformatica, 2002, 6(1): 7−34.

[19] Lee T, Moon B, Lee S. Bulk insertion for R-trees by seeded clustering. Data & Knowledge Engineering, 2006, 59(1): 86−106.

[20] Sander J, Ester M, Kriegel HP, Xu X. Density-based clustering in spatial databases: The algorithm GDBscan and its applications. Data Mining and Knowledge Discovery, 1998, 2(2): 169−194.

[21] Kondrak G. N-gram similarity and distance. In: Proc. of the 2005 Int'l Symp. on String Processing and Information Retrieval. 2005. 115−126.

[22] Ukkonen E. Approximate string-matching with *q*-grams and maximal matches. Theoretical Computer Science, 1992, 92(1): 191−211.

[23] Gravano L, Ipeirotis PG, Jagadish HV, Koudas N, Muthukrishnan S, Srivastava D. Approximate string joins in a database (almost) for free. Proc. of the VLDB Endowment, 2001, 1: 491−500.

[24] Christen P. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In: Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2008. 151−159.

[25] Ardalan A, Doan A, Akella A. Smurf: Self-service string matching using random forests. Proc. of the VLDB Endowment, 2018, 12(3): 278−291.

[26] Ramadan B, Christen P, Liang H, Gayler RW, Hawking D. Dynamic similarity-aware inverted indexing for real-time entity resolution. In: Proc. of the 2013 Pacific-Asia Conf. on Knowledge Discovery and Data Mining. 2013. 47−58.

[27] Altwaijry H, Mehrotra S, Kalashnikov DV. Query: A framework for integrating entity resolution with query processing. Proc. of the VLDB Endowment, 2015, 9(3): 120−131.

[28] Dragut EC, Ouzzani M, Elmagarmid AK. Query-time record linkage and fusion over Web databases. In: Proc. of the 31st IEEE Int'l Conf. on Data Engineering (ICDE). 2015. 42−53.

[29] Jagadish HV, Koudas N, Srivastava D. On effective multi-dimensional indexing for strings. ACM SIGMOD Record, 2000, 29(2): 403−414.

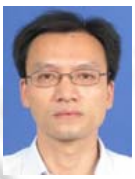
[30] Mudgal S, Li H, Rekatsinas T, Doan A, Park Y, Krishnan G, Deep R, Arcaute E, Raghavendra V. Deep learning for entity matching: A design space exploration. In: Proc. of the 2018 Int'l Conf. on Management of Data. 2018. 19−34.

附中文参考文献:

[2] 孙琛琛, 申德荣, 李玉坤, 肖迎元, 马建红. 时间约束的实体解析中记录对排序研究. 软件学报, 2020, 31(3): 695−709. http://www.jos.org.cn/1000-9825/5900.htm [doi: 10.13328/j.cnki.jos.005900]

[15] 孙琛琛, 申德荣, 寇月, 聂铁铮, 于戈. 面向关联数据的联合式实体识别方法. 计算机学报, 2015, 38(9): 1739−1754.

孙琛琛(1987－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为实体解析, 异常检测, 大数据分析与挖掘.



肖迎元(1969－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为数据库, 个性化推荐系统, 大数据挖掘与分析.

申德荣(1964－), 女, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为分布式数据管理, 数据集成.

李玉坤(1969－), 男, 博士, 教授, 主要研究领域为数据库, 信息检索,个人信息管理.