集合和字符串的相似度查询

林学民[1),2)] 王 炜[2)]

[1)](华东师范大学软件学院 上海 200062)

[2)](澳大利亚新南威尔士大学计算机科学与工程系 悉尼 2052)

摘 要 相似度查询是计算机学科中一个重要的问题,它的应用遍及多个领域,例如数据库、数据集成、互联网、数据挖掘以及生物信息学等.该文主要讨论在集合和字符串上的相似度查询.学术界从 2000 年来在这个领域内取得了大量的进展.作者总结了主要工作,并给出了作者的分析和归类.最后,该文提出了一些未来工作的方向.

关键词 相似度查询;相似度连接;前缀过滤;jaccard;编辑距离

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01853

Set and String Similarity Queries: A Survey

LIN Xue-Min[1),2)] WANG Wei[2)]

[1)](*Software Engineering Institute, Eastern China Normal University, Shanghai* 200062)

[2)](*School of Computer Science and Engineering, University of New South Wales, Sydney* 2052, *Australia*)

Abstract Similarity queries are an important topic in many domains in computer science, including databases, data integration, World Wide Web, data mining, and bioinformatics. In this paper, we focus on the subfield of similarity queries for sets and strings. There has been a wide body of development in this area since 2000. We survey major results and present our analyses and categorization of existing approaches. Finally, we conclude the paper by giving a list of future research directions.

Keywords similarity query; similarity join; prefix filtering; jaccard; edit distance

1 引 言

给定一组在数据库中的对象 R,一个查询对象 Q,一个相似度函数和相似度阈值 t,Q 的相似度查找(Similarity Search)会返回 R 中和 Q 相似度至少是 t 的对象.如果将另一组对象 S 中每一个对象作为 Q 进行相似度查询,其结果是 R 和 S 中相似度至少是 t 的对,这个操作叫作相似度连接(Similarity Join).我们将这两类查询统称为相似度查询(Similarity Queries).相似度查询在计算机科学的多个领域内有广泛的应用.例如,在记录用户兴趣的数据库中,每个用户关联了一组他或她感兴趣的网址.通过使用 Jaccard 相似度①,我们可以找出兴趣十分类似的用户[1].在某些应用中,我们更自然地可以使用距离函数.为了找到相似度高的对象组,我们可以限定距离不超过某个阈值.例如,为了集成多个客户数据的列表,我们可以把客户信息间编辑距离不超过某个阈值的客户对返回,由专人或者自动判别工具来判断它们是否是同一个客户[2].其它的典型的应用还包括:

(1)在一个记录了众多不同类型的地标的空间

收稿日期:2011-08-12;最终修改稿收到日期:2011-09-19.本课题得到澳大利亚研究理事会 Discovery Projects (DP110102937, DP0987557, DP0987273, DP0881035, DP0881779)、国家自然科学基金(NSFC61021004)及 Google 的资助.**林学民**,男,1961 年生,教授,主要研究方向为图数据库、不确定数据库、空间时间数据库以及相似度查询.E-mail: lxue@cse.unsw.edu.au. **王 炜**,博士,高级讲师,主要研究方向为相似度查询、数据库和信息检索的集成、查询处理和优化等.E-mail: weiw@cse.unsw.edu.au.

① 两个集合 A 和 B 的 Jaccard 相似度为 $|A\cap B|/|A\cup B|$.

数据库中,我们可以找出例如在地铁站附近(直线或者行走距离)500 m 内的意大利餐馆.

(2)在数据集成应用中,我们可以使用连接字段的值大致匹配的条件连接(Soft Join)多个异构的数据集[3-4].类似的应用还包括基于字典的命名实体识别(Dictionary-based Named Entity Recognition, DNER),例如在网页中自动标注产品名称.我们可以使用 Jaccard 相似度函数进行近似匹配(例如将"canon eos 5d digital slr camera"匹配到"canon eos 5d digital camera"[5]).

(3)在生物信息学中,我们需要近似的蛋白质或基因序列的匹配,以支持同源查找(Homology Search)[6]和下一代基因测序(Next-generation Sequencing)[7]等任务.

在本文中,我们主要关注基于集合和字符串的相似度连接(即,数据和查询对象都是集合和字符串).对于这个问题,主要的数据库领域内的工作是从 1997 年开始[2,8-9]①,并取得了许多重要的进展.本文将对其中主要的工作进行分门别类的介绍和分析,并给出了我们认为未来的有希望的工作方向.

本文第 2 节介绍问题的定义和相关的背景知识;第 3 节介绍集合上的相似度查询的技术和工作;第 4 节介绍字符串上的相似度查询的技术和工作;第 5 节介绍相关工作;第 6 节介绍我们对未来工作方向的展望;最后第 7 节进行总结.

2 问题定义和背景知识

给定两组对象 R 和 S,并定义在两个对象之间的相似度函数 sim 及其阈值 t,相似度连接的定义是返回所有 R 和 S 的对象中其相似度至少是 t 的那些对,即 $\{(r,s)\mid sim(r,s)\geq t, r\in R, s\in S\}$.为了表述方便,在本文中,我们仅考虑相似度自连接,即 $\{(r_1,r_2)\mid sim(r_1,r_2)\geq t, r_1\in R, r_2\in R\}$.在某些应用中,我们可以用距离函数代替以上定义中的相似度函数,并将判断条件"$sim(r_1,r_2)\geq t$"改为"$dist(r_1,r_2)\leq t$".

2.1 相似度函数阈值条件间的关系和转换

2.1.1 基于集合的相似度函数

给定两个集合 A 和 B,许多常见的基于集合的相似度函数的约束条件大多可以转化为等价的基于集合交的大小,即 $|A\cap B|$ 的约束条件.例如,Jaccard 的定义是两个集合的交的大小除以两个集合并的大小,而集合并的大小是两个集合各自大小减去它们交的大小.

定理 1. $Jaccard(A,B)\geq t$ 的充要条件是 $|A\cap B|\geq (t/(1+t))\times(|A|+|B|)$.

推论 1. $Jaccard(A,B)\geq t$ 的必要条件是 $|A\cap B|\geq t\times|A|$.

证明. 如果 $Jaccard(A,B)\geq t$ 则 $|B|\geq t\times|A|$,带入不等式右边,即可得到以上结论. 证毕.

由于以上这些约束条件完全等价,我们可以集中优化基于集合相交大小的约束,而不需要为每一个相似度函数做优化.

2.1.2 基于字符串的相似度函数

编辑距离是常见的衡量字符串相似度的函数之一.两条字符串 A 和 B 之间的编辑距离(记作 $ed(A,B)$)是使用插入、删除、替换操作将一条字符串改为另一条的所需的最少步数.字符串的编辑距离可以使用动态规划(dynamic programming)来计算,其时间复杂度为 $O(n\times m)$(n 和 m 分别为两条字符串的长度)②.

由于编辑距离的计算已经比较费时,在相似度查询中,如果我们还需要比较每一对字符串的话,其运行效率会非常低下.在实际的应用中,绝大多数的字符串对不满足阈值的限制.所以,我们可以通过使用一些过滤条件来剔除绝对不可能满足某个编辑距离的约束的字符串③.常见的过滤方法是将编辑距离的约束转换成特征(signature)集合间交的约束[2].常见的特征提取的方法是基于 q-gram,即提取该字符串 S 中所有的长度为 q 的子串(称作 grams(S)).grams(S)有 $|S|-q+1$ 个 q-gram④.

定理 2. $ed(A,B)\leq t$ 的必要条件是 $|\text{grams}(A)\cap\text{grams}(B)|\geq\max(|A|,|B|)-q+1-t\times q$.

证明. 可以证明在最坏情况下,每个编辑(插入、删除或者替换)操作一定可以破坏 q 个 q-gram.并且注意到字符串 X 生成 $|X|-q+1$ 个 q-gram.通过考虑从 $A\sim B$ 和 $B\sim A$ 的编辑操作,以上定理可证. 证毕.

---

① 在其它领域有更早的工作,例如文献[10].

② 文献中存在多种有更好的复杂度的计算方法,但实现则较为复杂.例如通过使用 Four Russians Techniques,编辑距离的计算可以在 $O(n\times\max(1,m/\log n))$ 时间内完成($n=m$)[11].通过仅仅计算必需的对角线上的元素,编辑距离的计算可以在 $O(d\times\min(m,n))$ 时间内完成[12].通过使用位运算的内在的并行性,编辑距离的计算可以在 $O(nm/w)$ 时间内完成(w 是机器的字长)[13].文献[7]提出对于字符集较小的情况下基于位操作的计算方法.

③ 这种计算模式称作先过滤再验证的模式(Filter-and-Verify).例如它在空间数据库中被广泛应用.

④ 也可以在字符串开头和/或结尾处加上不在字符集中的特殊字符(如).如果这样,生成的 q-gram 的个数以及必须相交的 q-gram 的个数需要做相应的调整.

文献[2]还考虑了 q-gram 的位置：只有两个相同的 q-gram 在各自字符串中的位置相差不超过阈值 t，我们才将它们算作匹配的 q-gram. 所以，以上定理的条件还可以增强为 matching_grams$(A,B)\geq\max(|A|,|B|)-q+1-t\times q$.

满足以上条件的字符串构成了候选集. 我们仅需对候选集中的字符串进行编辑距离的验证，从而能比每条字符串都验证更加高效. 这对于在关系数据库上支持基于编辑距离的查询意义尤其重大，因为以上的过滤条件都可以通过 SQL 来完成，仅仅最后的验证需要用户自定义函数(UDF)来实现.

3　集合的相似度查询

3.1　前缀过滤(Prefix Filtering)

根据以上相似度函数阈值条件间的转换，我们知道我们需要处理两个集合的交的大小大于等于一个阈值 t 的情况. 给定一个集合 Q，如何才能找到和 Q 相交大于等于 t 的集合呢?

传统的做法是：先找到一组候选集 C，即满足 Q 和 C 中任意元素 C_i 的交大于等于 1；然后在候选集中进行校验(Verification)，即计算 Q 和 C_i 的交的大小并和阈值 t 进行比较. 第一步可以使用倒排索引(Inverted Index)快速地获得候选集，即 $C=\{I(w)\mid w\in Q\}$，$I(w)$ 是元素 w 对应的倒排列表(Inverted List or Postings List). 该方法在实际数据集上的主要缺点是当 Q 中有一个元素出现的频率较高时，候选集会变得很大；恰恰在实际数据集上，元素的出现通常会遵循 Zipf 定律，所以有相当高的概率，Q 中会有一个高频的元素.

最早意识到该问题并提出解决方案的是文献[14]. 该论文提出了仅使用 Q 中最少出现的 $|Q|-t+1$ 个元素的倒排列表来获得候选集. 之后，文献[15]将这个想法进一步推广到数据，即所有数据集合 $|S|$ 也只要索引其 $|S|-t+1$ 个元素. 为了达到这一点，其附加的要求是所有集合的元素要按照同一个全局的顺序排序，并且我们仅考虑前 $|S|-t+1$ 个元素. 为了方便起见，我们把按照某种全局顺序排列过的集合的前 L 个元素叫做这个集合的 L-前缀. 我们可以定义以下的前缀过滤定理.

定义 1(前缀过滤). 两个集合 A 和 B 的交的大小大于等于 t 的必要条件是集合 A 的 $(|A|-t+1)$-前缀和集合 B 的 $(|B|-t+1)$-前缀的交大于等于 1.

前缀过滤应用到相似度自连接时，可以通过对需要连接的关系进行排序，从而得到更紧的过滤条件[16]. 这是因为我们可以限制每个查询的集合 Q 仅和比它小的集合进行比较. 以 Jaccard 为例，我们可以对 Q 取 $(|S|-\lceil t\times|S|\rceil+1)$-前缀，而对数据集中的每个集合 S 取它们的 $(|S|-\lceil 2t\times|S|/(1+t)\rceil+1)$-前缀. 其它的对前缀过滤的改进包括：

(1) 有权重的情况. 以上的前缀过滤规则可以扩展到元素有权重的情况[15].

(2) 延长前缀的长度. 在标准的前缀上在加入 k 个元素，与此同时要求候选集中的对象在延长的前缀中有 $k+1$ 个相同的元素. 这个方法可以有效地降低最终候选集的大小，但是因为前缀的增长，会要求读取更多的倒排列表，并必须考虑更多的对象是否会进入候选集. 其实际效果如何，很有可能非常依赖于数据集及程序的实现.

(3) 对倒排索引中无效的部分的处理[17]. 当一个集合 (A) 的前缀中的元素被索引后，它将一直留在索引里，并且以后任何一个前缀中包含该元素的字符串 (B) 都会把该记录 (A) 加入它的候选集，即使 A 和 B 的长度已经不符合大小过滤的条件. 因为在相似度连接时，我们用来做为查询集合的大小是递增的，根据当前查询集合的大小，我们可以动态地将倒排列表的前端那些不符合大小过滤条件的项删除.

(4) 全局顺序的选择. 影响候选集的大小的一个重要因素是出现在查询前缀中的元素的频度. 为了减少候选集的大小，一种通用的启发式方法(heuristic)是选择将所有元素按照它们的 IDF(Inverse Document Frequency)降序排列[14-15]. 这样会导致全局最不常见的元素出现在前缀中，从而帮助减少候选集的大小.

(5) 在数据库系统中实现和应用前缀过滤. 文献[16]考虑了在关系数据库上支持相似度查询的多种方法.

(6) 对倒排索引的改进. 在海量数据的情况下，对数据集建立的索引也会很大. 文献[18]考虑了不索引某些元素和将多个元素对应的倒排列表融合的方法，从而可以有效地减少索引的大小，使得我们更有可能在内存中装下索引. 文献[19]则考虑了索引远远大于内存、必须读取硬盘的情况下，内存中和硬盘上索引的物理组织和对应的查询处理的问题.

在文献中还存在几种不同的基于倒排索引来计算与查询集合交大于等于一个阈值的对象. 我们将查询集合 Q 中每个元素对应的倒排列表 L_i 来标识，

每个 L_i 包含了一组有序的数据集合的 ID. 我们可以简单的把这些 L_i 归并起来,同时保持对每个 ID 出现次数的计数,并返回计数值不小于阈值的 ID (即文献[20]中的 ScanCount 算法). 文献[21]研究了是否可以不需要将每个 L_i 都读取完就可以完成计算的问题,并给出了基于 Fagin 的 NRA 算法的改进算法. 文献[20]提出了另外几种优化的归并算法,包括考虑了通过利用相交的阈值和在倒排列表上的跳跃(Skipping)的方法达到仅仅读取一部分倒排列表的目的的 MergeSkip 算法以及将 MergeSkip 和前缀过滤的变种最优混合而得到的 DivideSkip 算法.

3.2 其它过滤的方法

3.2.1 大小过滤[2,15].

如果两个集合比较相似,它们的大小也应该比较接近. 对于仅仅是集合交的约束条件,大小过滤无法使用,因为即使两个集合很不相同,只要它们足够大,它们的交的大小也有可能会大于等于 t. 而大部分应用会避免以上情况的出现,从而选择使用某些方法归一化,例如除以集合的并的大小(Jaccard)或者两个集合大小的均值(Dice). 以 Jaccard 为例,大小过滤条件为 $t\times|A|\leq|B|\leq|A|/t$;对于编辑距离,其条件为 $|A|-t\leq|B|\leq|A|+t$.

3.2.2 使用位置信息[22]

我们注意到前缀过滤实质上是将集合变成了一个有序的序列. 我们通过使用元素在该序列中的位置信息①,就可以进一步增强前缀过滤的效果,以进一步减少候选集的大小. 给定一个元素 e,一个有序的元素序列 A 可以被分成左部和右部(分别记作 $A_l(e)$ 和 $A_r(e)$)并且保证 $A_l(e)$ 中所有元素严格(在全局顺序中)小于 e,$A_r(e)$ 中所有元素大于等于 e. 显然 $|A\cap B|=|A_l(e)\cap B_l(e)|+|A_r(e)\cap B_r(e)|$. 而 $|A_r(e)\cap B_r(e)|\leq\min(|A_r(e)|,|B_r(e)|)$. 如果我们将 e 在 A 和 B 中的位置记作 $\mathrm{pos}(A,e)$ 和 $\mathrm{pos}(B,e)$,我们可以得到以下的基于位置的过滤方法.

定理 3(位置过滤). 两个集合 A 和 B 的交的大小大于等于 t 的必要条件是对于任意元素 e,$|A_l(e)\cap B_l(e)|+1+\min(|A|-\mathrm{pos}(A,e),|B|-\mathrm{pos}(B,e))\geq t$.

推论 2. 两个集合 A 和 B 的交的大小大于等于 t 的必要条件是对于它们 $(|A|-t+1)$-前缀和 $(|B|-t+1)$-前缀中最小的相交的元素 e_{first},$1+\min(|A|-\mathrm{pos}(A,e_{\mathrm{first}}),|B|-\mathrm{pos}(B,e_{\mathrm{first}}))\geq t$.

证明. 因为 e_{first} 是前缀中最小的相交的元素,所以 $|A_l(e_{\mathrm{first}})\cap B_l(e_{\mathrm{first}})|=0$. 然后从前缀过滤条件和位置过滤条件可推得. 证毕.

3.2.2 后缀过滤[22]

在阈值较高的情况下,前缀过滤十分快捷,这主要是因为只有集合的一小部分被索引和读取. 例如:对 Jaccard 而言,索引和读取部分几乎是字符串长度的 $(1-t)$ 倍. 这同时意味我们考虑候选集时没有考虑字符串的(相当长的)后缀中可能出现的不匹配的元素. 后缀过滤就是通过进一步利用后缀中元素的位置信息,来进一步过滤前缀过滤不能排除的候选对. 其基本思想是选取一个有序集合的中位元素 e,然后获得两个集合的关于 e 的左部和右部. 可以证明对于符合 Jaccard 约束的两个集合的左部和右部的大小的差异也必定小于等于和 t 有关的一个函数的值. 我们可以选择用递归的方式多次使用后缀过滤,以达到更好的过滤效果.

4 字符串的相似度查询

4.1 基于集合的字符串的相似度查询

在这种查询中,我们将字符串先转化成集合,然后使用度量集合相似度的函数进行查询. 这类方法的优点(同时也是缺点)是不考虑字符之间的顺序. 以抄袭检测为例,这类方法可以有效地找出更换段落或者句子顺序的抄袭.

受篇幅所限,我们仅考虑 Jaccard 相似度的情况. 其它相似度函数的情况可以同理推出. 我们首先需要将字符串转化为集合. 常见的方法有每个元素是一个 token,或者是一个 q-gram. 然后,两条字符串的相似度可以用它们对应的集合的 Jaccard 来衡量. 所以我们之前讨论的对集合的相似度查询的处理方法可以直接使用.

4.2 基于字符串的相似度查询

在这种查询中,我们考虑基于字符串的相似度,即我们考虑字符间的顺序. 以抄袭检测为例,这类方法可以有效地找出仅有少量改动的大段抄袭.

下面我们将已有的方法分类进行介绍.

4.2.1 基于 q-gram 的方法

文献[2]给出了将字符串上的基于编辑距离的相似度连接转化成对应 q-gram 集合上的基于交的相似度连接. 我们可以使用以下的优化:先使用前缀过滤(前缀的长度为 $t\times q+1$),在得到的候选集上再判别是否有足够多的共同的(也成为匹配的) q-gram

① 位置从 1 开始. 另外,请注意这里讨论的位置不是该元素在其字符串中的位置.

(也称作数量过滤(Count Filtering))，最后，在剩下的候选集中，对每条字符串计算其与查询字符串的编辑距离[1].

由于将编辑距离的约束转化为基于不匹配的 q-gram 的过滤，已有的方法都是基于匹配的 q-gram，那么，不匹配的 q-gram 是否可以被利用起来呢？文献[23]提出了两种基于不匹配的 q-gram 的过滤. 第 1 种是通过考察在 $(t\times q+1)$-前缀中的 q-gram 的位置来进一步减少前缀的长度. 我们假定取 L-前缀 $(L<t\times q+1)$. 我们可以考察最少要多少个编辑操作才能将这 L 个元素全部破坏. 很明显，如果这 L 个元素互相没有重叠，那答案就是 l. 可以证明如果有重叠的话，通过排序后的一个线性的贪心算法就可以等到答案. 如果该结果大于编辑距离的阈值 t，我们就可以只使用这 L 个元素作为前缀. 通过两分查找，我们能够找到最小的 L-前缀满足至少需要 $t+1$ 个编辑操作才能将它们都破坏掉；我们把这样的前缀叫做最短前缀，它的取值范围在 $[t+1, t\times q+1]$ 之间.

第 2 种利用不匹配的 q-gram 的过滤方法是基于以下的观察：如果有一系列相邻的不匹配的 q-gram，那么这一区间内很有可能有较多的编辑操作. 我们可以通过对区间内子串建立字符集大小的向量，并测试对应向量的 ℓ_1 距离是否大于 $2t$.

4.2.2　基于变长的子串的方法

文献[24]提出了 Winnowing 的特征值提取的方法，相比以前的类似的方法(例如：文献[25-26])，这个新方法具有“局部”的特性，粗略地说，即特征的提取只决定于其周围局部范围内子串的内容. 对于用户给定的长度参数 L，通过调整参数，Winnowing 可以保证检测到两条字符串中有所有长度大于等于 L 的相同的子串(因为它们会产生相同的特征值). 利用这个特性，通过使用将字符串分治的方法，我们可以用 Winnowing 来支持编辑距离的相似度查询①.

另一种产生变长的子串的方法是 VGRAM[28-29]. 传统的 q-gram 方法的一个问题在于它要求索引所有长度为 q 的子串；由于子串的频率往往是不均匀分布的，这样会导致某些子串对应的倒排列表很长，从而导致算法需要付出不少的读取倒排列表的时间，并且候选集会较大. VGRAM 方法放宽了对 gram 长度的限制，它会选取长度在 $\lceil q_{\min}, q_{\max}\rfloor$ 之间的“优质”的 vgram，粗略来说，即对应的倒排列表较短的 vgram. 文献[28]提出了几种选取优质的 vgram 的方法，文献[29]则提出了使用动态编程的方法计算出两条相似的字符串需要共享的 vgram 个数的一个较紧的下界.

还有一种方法是 vchunk[27]. 其基本思想是将字符串分割成不重叠的子串(称作 vchunk). 并非所有的分割方法都可以考虑. 这是因为在编辑过程中因为插入和删除，可能会出现非常相似的字符串分割后的 vchunk 没有相同的情况②. 文献[27]提出了根据尾部受限模式字典(Tail-restricted Dictionary)分割的办法，可以保证每个编辑操作只会影响至多 2 个 vchunk. 在使用了前缀过滤，不匹配的 vchunk 的过滤和其它针对 vchunk 的过滤优化之后，我们只需要索引和读取 l-前缀即可 $(l\in[t+1, 2t+1])$.

4.2.3　基于枚举的方法[30]

一种最简单的支持编辑距离的方法是枚举所有和查询字符串的编辑距离小于等于 t 的所有字符串. 我们仅需要对数据库中的字符串建立 Hash 索引. 这个方法的最大问题是我们需要枚举 $O(|Q|^t|\sum|^t)$ 次. FastSS[31] 提出 t-删除邻居集(t-Deletion Neighborhood)的概念：即所有删除不超过 t 个字符所得到的字符串的集合. 两个字符串的编辑距离不超过 t 的必要条件是它们的 t-删除邻居集有共同的元素. 这个方法的查询运行时间为 $O(|Q|^t)$，即和字符集大小无关. FastSS 在字符串短和 t 极小时是十分快捷的(尤其是 $t=1$ 时，是线性的). 但是随着 t 的增加，其时间和空间复杂度都以指数级上升. 文献[32]通过适当的分治的方法，将数据库中的字符串和查询字符串都分片，并保证所有分片中至少有一份的编辑距离不能超过 1. 由于查询字符串和数据字符串长度可能不一样，查询字符串的每个分片还需要先在一定范围内作移动和伸缩，然后再生成 1-删除邻居集.

使用 t-删除邻居集的主要原因是为了避免考虑所有可能的替换操作. 让我们先考虑一个极其简单的特殊情况，即：(1) 我们不允许插入和删除操作(即所有字符串长度都是 l)；(2) 字符集的大小是 2(即 0 和 1). 这时问题其实变成了在二值向量集合上的基于 Hamming 距离的相似度连接，每个编辑操作就是将某一位置的值取反，阈值仍然为 t. 这时，我们仍然需要枚举 $C(|Q|, t)=O(|Q|^t)$ 个被编辑过的位置的组合.

很明显，如果我们可以减小 l 的话就可以降低代价. 我们可以将向量划分成 k 段 $(k>t)$，至少有 $k-t$ 段没有被编辑过. 我们可以枚举所有 $C(k,t)$ 个

① 具体的细节请参见文献[27].

② 事实上，编辑距离的难点之一就是允许插入和删除.

编辑过的段,在每种情况下,使用没有被编辑过的段的位置和内容作为查询条件找到数据库中匹配的字符串,这些字符串的并构成了候选集.在候选集上进行校验就可以获得最后的结果.这种方法产生的候选集依赖于参数 k:k 越小,枚举的情况就越少,但是对应的精确匹配的条件就越松.文献[33]讨论了在海量 64 位二值向量上为了支持 $t=3$ 的基于 Hamming 距离的查询的几种参数设定的方法.这个思路也被应用在文献[7]中.由于文献[7]处理的问题是在一条很长的字符串上做基于编辑距离的近似子串匹配,以上方法实质上是索引了 $C(k-1,t)$ 种不同形状的带间隔的子序列.

以上方法还有另一种变种.我们可以再多使用一层划分.根据抽屉原理,如果我们先将向量划分成 n_1 段,则至少存在一段,在这段上的编辑操作至多是 $t_1=\lfloor t/n_1 \rfloor$.然后,我们在每一段上再使用划分+枚举的方法[33].这就是 PartEnum 算法[34].该算法的优点在于由于 t_1 的向下取整运算,阈值实际上被降低了.该算法需要手工调节两个划分的参数,在实际中,仅当 t 较小时,通过手工调参能够取得比较优秀的结果.

最后,我们给出从基于编辑距离的连接转换成基于海明距离的连接的方法.我们将所有数据字符串转成它们对应的 q-gram 的集合.如果在同一条字符串中有多个相同的 q-gram(例如 ab),我们把它们出现的顺序作为它们的下标(例如 ab_1,ab_2 等),从而认为它们是不同的元素[15].把每个出现过的(带下标的)q-gram 作为一个维度,那么每个 q-gram 集合就可以转化成一个二值向量.

定理 4. 两个字符串的编辑距离不超过阈值 t 的必要条件是它们对应的 q-gram 集合对应的二值向量的 Hamming 距离不超过 $q\times t$[34].

最近,文献[7]提出了位操作来快速判断两条二进制的字符串的编辑距离是否小于等于 1.为了支持编辑距离小于等于 t 的情况,文献[7]枚举所有不同的 t 个编辑操作的排列(permutation);在依次对一条二进制字符串使用某个编辑操作的排列时,一旦中间结果和另一条字符串相等①,则可以判断它们的编辑距离小于等于 t.

4.2.4 一个统一的基于特征的方法[30]

以上所有方法都可以归结到以下的算法框架中:我们(1)为数据字符串生成 λ_t 个特征值(并把它们加入索引);(2)为查询字符串生成 Λ_t 个特征值;(3)候选集是那些和查询字符串共享至少 LB_t 个特征值的数据字符串.我们可以用 $\Upsilon(\lambda_t,\Lambda_t,LB_t)$ 来描述一个方法.例如,传统的基于 q-gram 的方法[2]使用的特征值是 q-gram,可以记作 $\Upsilon(|S|-q+1,|Q|-q+1,\max(|S|,|Q|)-q+1-q\times t)$.而使用了前缀过滤后的基于 q-gram 的方法[23],可以记作 $\Upsilon(q\times t+1,q\times t+1,1)$.文献[30]中给出了对更多方法的刻画.

这个框架简洁地刻画出不同方法的空间和时间的复杂度,并且这些复杂度随阈值 t 的变化趋势.另外,如果我们把完全相同的匹配查询看作是基于编辑距离相似度查询的一个特例,即 $t=0$,那么,基于 Hash 表的查询方法可以看作使用整个字符串作为一个特征值,可以记作 $\Upsilon(1,1,1)$.

一个自然的问题就是数据或者查询字符串最少要生成多少个特征值.文献[30]证明了最少需要 $t+1$ 个特征值,并给出了两个算法,分别在数据和查询字符串使用 $t+1$ 的特征值.其主要思想是在一条字符串上提取所有的 q-chunk(不足时补特殊字符),即相邻的,不相交的长度为 q 的子串,而在另一条字符串上取所有的 q-gram.可以证明每个编辑操作最多破坏一个 q-chunk.结合前缀过滤,我们可以获得 $\Upsilon(t+1,|Q|-(\lceil(|Q|-t)/q\rceil-t)+1,1)$ 的 IndexChunk 算法或者 $\Upsilon(|S|-(\lceil(|S|-t)/q\rceil-t)+1,t+1,1)$ 的 IndexGram 算法.这两个算法不仅在绝大多数情况下比已有的方法更快,而且展现的不同的时间和空间的特性.例如:IndexGram 算法仅仅需要读取 $t+1$ 条倒排列表,所以其查询速度非常快,但这也要求所有的 q-gram 需要被索引,导致索引的大小会较大.

4.2.5 基于 Trie[35-36] 和树[37] 的方法

Trie 是一种有效的存储多条字符串的数据结构,它支持快速的查找,并且通过在字符串之间共享前缀来节省空间.因为前缀共享,Trie 可以很好地支持基于编辑距离的查询:当计算查询字符串和两条共享长度为 L 的前缀的字符串的编辑距离时,它们的编辑距离矩阵的前 L 列是相同的[38].文献[36]利用 Trie 来支持增量相似度查询,文献[35]利用 Trie 来进行相似度连接.由于以上的特性,Trie 适合用于数据集的字符串较短的情况.

B^{ed}-Tree 是一个基于 B+树的管理外存中字符串集并支持多种基于(归一化)编辑距离的查询(范围查询,top-k 查询,和连接)[37].其关键步骤是选取一种将字符串排列的顺序,使得给定任何一段连续的字符串区间,都可以快速地得到这个区间内字符

① 可以使用异或操作(XOR)快速地判断.

串和查询字符串的编辑距离的下界.

5　相关工作

更多的相关工作的介绍可以参见近期的辅导报告和综述[39-45].

5.1　早期的相似度查询的相关工作

早期的相似度连接的工作主要集中在对象为欧几里得空间(Euclidean Space)中的点,而且使用的是 ℓ_2 的距离函数,这个问题通常被称作空间连接(Spatial Join).在低维空间的工作主要有循环、平面扫描(Plane Sweeping)、空间填充曲线(Space Filling Curve)、基于一颗或两颗 R-tree 以及分治的方法.具体的介绍和比较可以参见近期的一个综述[46].最近,文献[47]考虑了利用 GPU 的高并行性和计算速度来实现快速的空间连接.当空间中的维数较高时,空间连接是一个很难的问题.一方面是因为维度灾难(Curse of Dimensionality),另一方面是因为高维空间中没有性能优异的索引.现有的方法主要基于空间的划分.例如,文献[48-50].

还有一些工作是在度量空间(Metric Space)中进行相似度连接.例如,使用度量空间的索引[51-52]或者使用分治的策略[53].

在集合和字符串上的常见的相似度和距离函数往往不能有效地映射到欧几里德空间或者度量空间上,因此它们的相似度查询需要新的方法.早期的工作主要考虑简单的二值的相似度函数,例如相交不为空以及包含.文献[8]考虑 $R \bowtie_{R.a \subseteq S.b} S$ 提出了使用对集合的特征值进行过滤的 Hash 连接算法.文献[54]提出了另一种基于 Hash 和复制的连接算法.文献[55]提出了使用倒排索引来支持相交不为空,或者包含约束的连接.

5.2　近似的相似度查询的方法

在文献中,有大量为相似度查询提供近似解的方法.其中最有代表性的是位置敏感 Hash(Locality Sensitive Hash,LSH)[56].给定一个相似度函数,这类方法为每个对象(集合或者字符串)生成为一组特征值,我们仅需对有相通特征值的对象进行验证.通过调节将几个 LSH 的特征值拼接在一起作为索引和查询的基本单位,可以达到加快查询速度的目的;通过调节将以上步骤重复的次数,可以保证最后达到一个用户制定的召回率的阈值[16,57].

(1) 对 Jaccard 来说,基本的 LSH 方法是 min-hash[58] 及其改进 b-bit min-hashing[59].

(2) 对 Cosine 来说①,基本的 LSH 方法是 sim-hash[33].

(3) 对 Hamming 来说,基本的 LSH 方法是 random projection.

对其它相似度或距离函数有效的 LSH 函数可以参见文献[60].

5.3　其它的问题定义

实体识别(Named Entity Recognition, NER)是另一个重要的应用:有一组字符串组成的字典,每个字符串是一个实体的名字.对于每一个输入的文件,我们需要找到所有近似匹配字典中某个实体的子串.实体识别往往是其它步骤和应用(例如文本检索和文本数据挖掘)的基础.文献[5,61]考虑了 Jaccard 约束下的实体识别,文献[32,62]考虑了编辑距离约束下的实体识别.文献[7]考虑的问题可以看作字典中仅有一项长度固定的实体的情况.

Top-k 相似度查询[63].在某些应用中,我们可能无法预先设定相似度函数的阈值.除了盲目地尝试不同的阈值以外,我们可以使用 top-k 相似度查询,即返回最相似的 k 对结果.文献[63]支持基于 Jaccard 的 top-k 查询,它使用了事件驱动的模型,动态地调整相似度的阈值,并且提出了防止重复校验的候选结果、简短倒排索引以及更紧的上界估计的技术.文献[37]提出的 B^{ed}-tree 可以有效地支持基于编辑距离的 top-k 查询.

Map-Reduce 上的相似度查询[64-66].以上的快速的相似度查询算法因为假设所有索引和数据都在内存中,当数据量增长到很大的数量级时,这会使得它们不能在单机上运行.鉴于基于 Map-Reduce 的分布式计算在其它海量数据计算中的成功,我们可以使用 Map-Reduce 的框架来支持大数据量的相似度查询.早期的工作[66]关注的是计算所有对象对之间的相似度(即没有阈值的限制).文献[64-65]都研究了有阈值的情况,并将单机版的一些过滤条件延伸到 Map-Reduce 框架上.

6　未来的工作

随着数据(尤其是互联网上和社会网路上的数据)的快速增长,越来越多的应用会涉及到集合和字符串的相似度查询.我们下面列出在这个领域内 3 个重要的未来工作的方向.

6.1　相似度查询的理论基础

一方面,绝大多数现有的方法是基于一种或者

① 精确的说是 arccos.

多种启发性的方法.我们对这些方法的代价、适用情况以及相互之间的关系还不甚了解.例如,前缀过滤在大量的、来自于不同领域的数据集上都取得了很好的效果.但是,如果许多低频的元素的出现频率仍然较高,或者相似度阈值过小或距离阈值过大时,它的过滤效果会迅速下降.如果没有一个理论上的分析,我们就无法预测何时这种过滤有效,也会阻碍我们寻找在前缀过滤失效的情况下,仍然有效的过滤方法①.现在,已经有不少工作集中在估计各种相似度查询的结果的大小或选择度(Selectivity)的问题上,包括文献[67-71].

6.2 相似度查询的方法的创新和改进

过去的研究工作将对于集合和字符串的相似度连接的效率提高了几个数量级.另外,不同的方法往往在某些参数范围内有良好的表现(可以参见文献[30]的试验结果).与此同时,数据量的增长和应用对时间及空间的不断严苛的要求都会促使我们继续寻找新的方法或者对已有方法进行改进.例如,文献[30]提出了使用不对称的特征值以降低索引大小或者查询时间的方法.我们可以研究这种思路是否可以进一步完善、调整,或应用到不同的相似度查询中.文献[35-36]使用 Trie 来存储字符串和共享中间计算结果,或者利用它来有效地支持增量式的查询.但是 Trie 本身有一些弱点,例如:其占用空间较大,对较长的或没有许多前缀共享的字符串集合效果就比较差.所以我们可以探究是否可以通过分块或者改进的 Trie 来进一步提高现有的方法的效率和适用范围.

6.3 支持新的相似度查询和应用

一方面,大多数现有的工作集中在常见的比较简单的相似度和距离函数上.在不同的引用中,我们可能会面临更多的更复杂的函数.最近的一些工作已经开始研究新的函数,例如将编辑距离和 Jaccard 结合[72]、Earth Mover's Distance[73]、Bregman Divergence[74]等.另一方面,对于集合和字符串的相似度连接的工作中提出的许多方法和技巧,可以应用到其它问题和领域内.例如:文献[75]使用相似度连接来对数据聚类;将基于编辑距离的近似字符串匹配和空间数据结合以支持多种新的查询[76];在有噪音的字符串上的近似查询[77];将近似字符串匹配和在数据库上的关键字检索结合[78]等.

7 结束语

相似度查询是一个基本问题,在多个不同的计算机领域都有应用.自 2000 年以来,学术界在对集合和字符串的相似度查询的问题上作了很多研究,并取得了许多进展.本文对该领域内的主要的有代表性的工作作了一个总结和整理,并且对未来的工作作了展望.

参 考 文 献

[1] Bayardo R J, Ma Y, Srikant R. Scaling up all pairs similarity search//Proceedings of the WWW. 2007

[2] Gravano L, Ipeirotis P G, Jagadish H V, Koudas N, Muthukrishnan S, Srivastava D. Approximate string joins in a database (Almost) for free//Proceedings of the VLDB. 2001

[3] Cohen W W. Integration of heterogeneous databases without common domains using queries based on textual similarity//Proceedings of the SIGMOD Conference. 1998: 201-212

[4] Cohen W W, Hirsh H. Joins that Generalize: Text classification using WHIRL//Proceedings of the KDD. 1998: 169-173

[5] Chakrabarti K, Chaudhuri S, Ganti V, Xin D. An efficient filter for approximate membership checking//Proceedings of the SIGMOD Conference, 2008: 805-818

[6] Papapetrou P, Athitsos V, Kollios G, Gunopulos D. Reference-Based alignment in large sequence databases//Proceedings of the PVLDB. 2009, 2: 205-216

[7] Li Y, Terrell A, Patel J M. WHAM: A high-throughput sequence alignment method//Proceedings of the SIGMOD Conference. 2011: 445-456

[8] Helmer S, Moerkotte G. Evaluation of main memory join algorithms for joins with set comparison join predicates//Proceedings of the VLDB. 1997: 386-395

[9] Baeza-Yates R A, Gonnet G H. A fast algorithm on average for all-against-all sequence matching//Proceedings of the SPIRE/CRIWG. 1999: 16-23

[10] Gonnet G H, Cohen M A, Benner S A. Exhaustive matching of the entire protein sequence database. Science, 1992, 256: 1443-1445

[11] Masek W J, Paterson M. A faster algorithm computing string edit distances. Journal of Computer and System Sciences, 1980, 20(1): 18-31

[12] Ukkonen E. Algorithms for approximate string matching. Information and Control, 1985, 64: 100-118

[13] Myers G. A fast bit-vector algorithm for approximate string matching based on dynamic programming//Proceedings of the CPM. 1998: 1-13

[14] Sarawagi S, Kirpal A. Efficient set joins on similarity predicates//Proceedings of the SIGMOD. 2004

[15] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning//Proceedings of the ICDE. 2006

① 在使用一系列假设后,我们初步的建模结果是:(仅仅)前缀过滤产生的基于 Jaccard 的相似度连接候选集大小对阈值的依赖关系是 $O((1-t)^3 t)$,即当 t 减小时,候选集会迅速增长.

[16] Xiao C, Wang W, Lin X, Yu J X, Wang G. Efficient similarity joins for near duplicate detection. ACM Transactions on Database Systems, 2011, 36(3): 1-41

[17] Ribeiro L A, Harder T. Generalizing prefix filtering to improve set similarity joins. Information Systems, 2011, 36(1): 62-78

[18] Behm A, Ji S, Li C, Lu J. Space-constrained gram-based indexing for efficient approximate string search//Proceedings of the ICDE. 2009: 604-615

[19] Behm A, Li C, Carey M J. Answering approximate string queries on large data sets using external memory//Proceedings of the ICDE. 2011: 888-899

[20] Li C, Lu J, Lu Y. Efficient merging and filtering algorithms for approximate string searches//Proceedings of the ICDE. 2008

[21] Hadjieleftheriou M, Chandel A, Koudas N, Srivastava D. Fast indexes and algorithms for set similarity selection queries//Proceedings of the ICDE. 2008: 267-276

[22] Xiao C, Wang W, Lin X, Yu J X. Efficient similarity joins for near duplicate detection//Proceedings of the WWW. 2008

[23] Xiao C, Wang W, Lin X. Ed-join: An efficient algorithm for similarity joins with edit distance constraints//Proceedings of the PVLDB. 2008, 1: 933-944

[24] Schleimer S, Wilkerson D S, Aiken A. Winnowing: Local algorithms for document fingerprinting//Proceedings of the SIGMOD Conference. 2003: 76-85

[25] Manber U. Finding similar files in a large file system//Proceedings of the USENIX Winter. 1994: 1-10

[26] Brin S, Davis J, Garcia-Molina H. Copy detection mechanisms for digital documents//Proceedings of the SIGMOD Conference. 1995: 398-409

[27] Wang W, Qin J, Xiao C, Lin X, Shen H T. VChunkJoin: An efficient algorithm for edit similarity joins. University of New South Wales, 2011

[28] Li C, Wang B, Yang X. VGRAM: Improving performance of approximate queries on string collections using variable-length grams//Proceedings of the VLDB. 2007

[29] Yang X, Wang B, Li C. Cost-based variable-length-gram selection for string collections to support approximate queries efficiently//Proceedings of the SIGMOD Conference. 2008: 353-364

[30] Qin J, Wang W, Lu Y, Xiao C, Lin X. Efficient exact edit similarity query processing with the asymmetric signature scheme//Proceedings of the SIGMOD Conference. 2011: 1033-1044

[31] Bocek E H T, Stiller B. Fast similarity search in large dictionaries. Department of Informatics, University of Zurich ifi-2007.02, 2007

[32] Wang W, Xiao C, Lin X, Zhang C. Efficient approximate entity extraction with edit constraints//Proceedings of the SIMGOD. 2009

[33] Manku G S, Jain A, Sarma and A D. Detecting near-duplicates for web crawling//Proceedings of the WWW. 2007: 141-150

[34] Arasu A, Ganti V, Kaushik R. Efficient exact set-similarity joins//Proceedings of the VLDB. 2006

[35] Wang J, Feng J, Li G. Trie-join: Efficient trie-based string similarity joins with edit//Proceedings of the VLDB. 2010

[36] Chaudhuri S, Kaushik R. Extending autocompletion to tolerate errors//Proceedings of the SIGMOD Conference. 2009

[37] Zhang Z, Hadjieleftheriou M, Ooi B C, Srivastava D. B^{ed}-tree: An all-purpose index structure for string similarity search based on edit distance//Proceedings of the SIGMOD Conference, 2010: 915-926

[38] Shang H, Merrett T H. Tries for approximate string matching. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(4): 540-547

[39] Koudas N, Sarawagi S, Srivastava D. Record linkage: Similarity measures and algorithms//Proceedings of the SIGMOD Conference. 2006: 802-803

[40] Zezula P, Amato G, Dohnal V, Batko M. Similarity Search: The Metric Space Approach. Springer, 2010

[41] Samet H. Similarity searching: Indexing, nearest neighbor finding. Dimensionality reduction, and embedding methods for applications in multimedia databases//Proceedings of the ICDE. 2009

[42] Hadjieleftheriou M, Li C. Efficient approximate search on string collections//Proceedings of the PVLDB. 2009, 2: 1660-1661

[43] Wang W. Similarity join algorithms: An introduction//Proceedings of the National Database Conference, China, 2010

[44] Navarro G. A guided tour to approximate string matching. ACM Computing Surveys, 2001, 33(1): 31-88

[45] Hadjieleftheriou M, Srivastava D. Approximate string processing. Foundations and Trends in Databases, 2011, 2(4): 267-402

[46] Jacox E H, Samet H. Spatial join techniques. ACM Transactions on Database Systems, 2007, 32(1)

[47] Lieberman M D, Sankaranarayanan J, Samet H. A fast similarity join algorithm using graphics processing units//Proceedings of the ICDE. 2008: 1111-1120

[48] Shim K, Srikant R, Agrawal R. High-dimensional similarity joins//Proceedings of the ICDE. 1997: 301-311

[49] Koudas N, Sevcik K C. High dimensional similarity joins: Algorithms and performance evaluation. IEEE Transactions on Knowledge and Data Engineering, 2000, 12(1): 3-18

[50] Dittrich J-P, Seeger B. GESS: A scalable similarity-join algorithm for mining large data sets in high dimensional spaces//Proceedings of the KDD, 2001: 47-56

[51] Dohnal V, Gennaro C, Zezula P. Similarity join in metric spaces using eD-Index//Proceedings of the DEXA. 2003

[52] Paredes R, Reyes N. List of Twin Clusters: A data structure for similarity joins in metric spaces//Proceedings of the SISAP. 2008: 131-138

[53] Jacox E H, Samet H. Metric space similarity joins. ACM Transactions on Database Systems, 2008, 33(2): 1-38

[54] Ramasamy K, Patel J M, Naughton J F, Kaushik R. Set containment joins: The good, the bad and the ugly//Proceedings of the VLDB. 2000: 351-362

[55] Mamoulis N. Efficient processing of joins on set-valued attributes//Proceedings of the SIGMOD Conference. 2003: 157-168

[56] Indyk P, Motwani R. Approximate nearest neighbors: Towards removing the curse of dimensionality//Proceedings of the STOC. 1998: 604-613

[57] Theobald M, Siddharth J, Paepcke A. SpotSigs: Robust and efficient near duplicate detection in large web collections//Proceedings of the SIGIR. 2008: 563-570

[58] Broder A Z. On the resemblance and containment of documents//Proceedings of the SEQS. 1997

[59] Li P, Konig A C. b-Bit minwise hashing//Proceedings of the WWW. 2010: 671-680

[60] Andoni A, Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM, 2008, 51(1): 117-122

[61] Agrawal S, Chakrabarti K, Chaudhuri S, Ganti V. Scalable ad-hoc entity extraction from text collections//Proceedings of the PVLDB. 2008, 1: 945-957

[62] Li G, Deng D, Feng J. Faerie: Efficient filtering algorithms for approximate dictionary-based entity extraction//Proceedings of the SIGMOD Conference. 2011: 529-540

[63] Xiao C, Wang W, Lin X, Shang H. Top-k set similarity joins//Proceedings of the ICDE. 2009: 916-927

[64] Wang C, Wang J, Lin X, Wang W, Wang H, Li H, Tian W, Xu J, Li R. MapDupReducer: Detecting near duplicates over massive datasets//Proceedings of the SIGMOD Conference. 2010: 1119-1122

[65] Vernica R, Carey M J, Li C. Efficient parallel set-similarity joins using MapReduce//Proceedings of the SIGMOD Conference. 2010: 495-506

[66] Lin J J. Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce//Proceedings of the SIGIR. 2009: 155-162

[67] Jin L, Li C, Vernica R. SEPIA: Estimating selectivities of approximate string predicates in large Databases. The VLDB Journal, 2008, 17(5): 1213-1229

[68] Lee H, Ng R T, Shim K. Extending Q-grams to estimate selectivity of string matching with low edit distance//Proceedings of the VLDB. 2007: 195-206

[69] Lee H, Ng R T, Shim K. Approximate substring selectivity estimation//Proceedings of the EDBT. 2009: 827-838

[70] Lee H, Ng R T, Shim K. Power-law based estimation of set similarity join size//Proceedings of the PVLDB. 2009, 2: 658-669

[71] Lee H, Ng R T, Shim K. Similarity join size estimation using locality sensitive hashing//Proceedings of the PVLDB. 2011, 4: 338-349

[72] Wang J, Li G, Feng J. Fast-join: An efficient method for fuzzy token matching based string similarity join//Proceedings of the ICDE. 2011: 458-469

[73] Zhang Z, Ooi B C, Parthasarathy S, Tung A K H. Similarity search on Bergman divergence: Towards non-metric indexing//Proceedings of the PVLDB. 2009, 2: 13-24

[74] Xu J, Zhang Z, Tung A K H, Yu G. Efficient and effective similarity search over probabilistic data based on earth mover's distance//Proceedings of the PVLDB. 2010, 3: 758-769

[75] Bohm C, Braunmueller B, Breunig M M, Kriegel H-P. High performance clustering based on the similarity join//Proceedings of the CIKM. 2000: 298-305

[76] Yao B, Li F, Hadjieleftheriou M, Hou K. Approximate string search in spatial databases//Proceedings of the ICDE. 2010: 545-556

[77] Jestes J, Li F, Yan Z, Yi K. Probabilistic string similarity joins//Proceedings of the SIGMOD Conference. 2010: 327-338

[78] Li G, Ji S, Li C, Feng J. Efficient type-ahead search on relational data: A TASTIER approach//Proceedings of the SIGMOD Conference. 2009: 695-706

LIN Xue-Min, born in 1961, professor. His research interests include graph databases, uncertain databases, spatial-temporal databases, and similarity query processing.

WANG Wei, born in 1977, Ph. D., senior lecturer. His research interests include similarity query processing, integration of database and information retrieval technologies, and query processing and optimization.

Background

In many areas in Computer Science, we digitalize an object (e. g., by a set of its feature values) and then evaluate the relationship between objects using the similarity or distance values between their digital representations. Similarity query processing, including both search and joins, aims to accelerate the process of retrieving similar objects. There has been a plethora of studies on this topic. While earlier studies mainly focus on low or high-dimensional Euclidean spaces, and metric spaces, we report recent development for objects that are modeled as sets or strings. Apart from surveying and categorizing representative work in this area, the main contribution is perhaps our own analyses on pros and cons of existing methods and their relationships. Finally, we give a list of future research directions.

This work is supported by Australian Research Council Discovery Projects 110102937, 0987557, 0881035, 0881779, NSFC 61021004, and Google Research Award.