软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(2):329−347 [doi: 10.13328/j.cnki.jos.004934]

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

空间关键词搜索研究综述*

刘喜平[1,2], 万常选[1,2], 刘德喜[1,2], 廖国琼[1,2]

[1](江西财经大学 信息管理学院,江西 南昌 330013)

[2](江西省高校数据与知识工程重点实验室(江西财经大学),江西 南昌 330013)

通讯作者: 刘喜平, E-mail: lewislxp@gmail.com, lxpq@qq.com

摘 要: 由于越来越多的数据具有位置和文本双重属性,空间关键词查询(spatial keyword query,简称 SKQ)应运而生.一个 SKQ 以一个地理位置和若干关键词作为参数,返回满足空间与文本约束的结果,这些结果往往根据指定公式排列.对现有的空间关键词搜索技术进行了梳理,首先对问题进行了描述,对挑战进行了分析;然后分析了基本空间关键词搜索技术.将文献中提出的各种空间关键词查询进行了划分,对现有的查询处理技术进行分类,对每种类型的技术,从索引技术和查询算法两个方面进行了总结,并从多个角度对它们进行了比较.其后介绍了扩展空间关键词搜索技术,还介绍了与该问题相关的其他研究工作.最后指出了研究中存在的不足以及以后的研究方向.

关键词: 空间关键词查询;索引;查询处理技术

中图法分类号: TP311

中文引用格式: 刘喜平,万常选,刘德喜,廖国琼.空间关键词搜索研究综述.软件学报,2016,27(2):329−347. http://www.jos.org.cn/1000-9825/4934.htm

英文引用格式: Liu XP, Wan CX, Liu DX, Liao GQ. Survey on spatial keyword search. Ruan Jian Xue Bao/Journal of Software, 2016,27(2):329−347 (in Chinese). http://www.jos.org.cn/1000-9825/4934.htm

Survey on Spatial Keyword Search

LIU Xi-Ping[1,2], WAN Chang-Xuan[1,2], LIU De-Xi[1,2], LIAO Guo-Qiong[1,2]

[1](School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330013, China)

[2](Jiangxi College Key Laboratory of Data and Knowledge Engineering (Jiangxi University of Finance and Economics), Nanchang 330013, China)

Abstract: As more and more data have both spatial and textual attributes, spatial keyword query (SKQ) has been proposed to enhance the data search. An SKQ takes a spatial location and a set of keywords as arguments, and returns objects satisfying spatial and textual constraints which are then probably ranked according to certain functions. This paper investigates the current spatial keyword search techniques. It first defines the problem and analyzes the challenges. Then, it discusses the basic spatial keyword query processing techniques. Specifically, it categorizes the proposed queries according to their components, and then classifies the existing query processing techniques into three groups. For each group, the paper reviews the proposed indexing and query processing techniques. The paper also compares these techniques from several perspectives. In addition, it discusses extensions to the basic SKQ. Finally, it addresses research work related to SKQ, as well as some future research directions.

Key words: spatial keyword query; index; query processing technique

* 基金项目: 国家自然科学基金(61363010, 61173146, 61262009, 61363039, 61462037); 江西省高等学校科技落地计划(KJLD12022); 江西省科技厅青年科学基金(20151BAB217005, 20142BAB217014); 江西省教育厅科技项目(GJJ12732)

Foundation item: National Natural Science Foundation of China (61363010, 61173146, 61262009, 61363039, 61462037); Science & Technology Landing Project of Jiangxi Province (KJLD12022); Science Foundation for Youths of Jiangxi Province (20151BAB217005, 20142BAB217014); Science & Technology Foundation of Department of Education of Jiangxi Province (GJJ12732)

收稿时间: 2015-05-22; 修改时间: 2015-09-10; 采用时间: 2015-10-17; jos 在线出版时间: 2015-11-03

CNKI 网络优先出版: 2015-11-04 17:10:14, http://www.cnki.net/kcms/detail/11.2560.TP.20151104.1710.013.html

随着无线通信技术和智能移动终端的广泛应用,地理位置与文本数据的融合越来越普遍.一方面,越来越多的 Web 对象具有位置属性,如景点、餐厅的网页中都含有地理位置,Tweet、微博可以自动带上位置信息.这一趋势使得 Web 具有了一个新的维度——空间维.另一方面,很多兴趣点(point of interest,简称 POI)具有文本描述,如网页、微博、微信中对一个兴趣点的描述除了其地理位置之外,还有文本;在签到应用中(如 foursquare),用户在签到的同时其位置坐标会被自动记录,与此同时,用户往往会用文字描述心情、感想.随着基于位置的服务(location-based service,简称 LBS)的飞速发展,这类空间-文本数据会越来越多[1].

空间查询和关键词搜索是搜寻空间对象和文本对象(如 Web 网页)的基本方式.由于空间-文本对象具有位置和文本双重属性,很自然地,可以将空间查询和关键词搜索结合起来,以实现对空间-文本对象的更有效的搜索.空间关键词搜索是空间查询和关键词搜索的结合,即用户的查询不仅包含关键词,还包含一个地理位置,而查询的结果则是满足空间和文本约束,并与查询相关(如文本相似、位置相近)的对象.

空间关键词搜索在实际生活中有很多应用.例如,一个旅行者在一个陌生的城市中可以发出一个空间关键词查询,以搜索离他最近的"中餐馆".在一个签到应用中,一个游客可以查询在他周围 5 公里之内的签到中哪些提到了"购物"、"特产"、"纪念品"等关键词.在 Flickr 等图片网站上,用户可以查询在某一景点中拍摄的,并且关于"日出"、"云海"的照片.在实际的搜索引擎中,有很多查询与空间关键词查询有关.2012 年的一份研究报告指出[2]:Google 处理的搜索中有 24%与获取本地信息有关,目的是为了搜索空间 Web 对象或地点.而根据另一份报告[3],Bing 搜索引擎中 56%的移动搜索是为了获取本地信息.

在学术界,这一技术已经引起了研究者的极大兴趣.从 2005 年左右至今,在数据库、信息检索、地理信息系统等领域发表了大量的研究论文,并提出了一系列新技术.本文将对这些研究进行梳理.本文从基本问题-扩展问题-相关问题这 3 个层次分析了相关技术.对于基本问题,文章首先区分了空间关键词查询的不同类型,然后对提出的查询处理技术进行了分类,对每种类型的技术,总结了其索引技术和查询算法,并从多个角度对各种查询处理技术的优缺点做了比较.对于扩展问题,总结了 3 个重要研究方向;对于相关问题,介绍了其主要思想.文章还列出了当前存在的问题和可能的研究方向.

1 问题与挑战

1.1 问题描述

在文献中,称既有地理位置属性又有文本属性的对象为空间-文本对象(spatio-textual object)[4].一个空间-文本对象 o 由两部分组成:地理位置 $o.loc$ 和关键词集合 $o.kws$,其中,$o.loc$ 一般表示为二维平面上的一个点.例如,一个酒店的网页和黄页网站中的一个商家既包括若干关键词,也包括地理位置,可以理解为一个空间-文本对象.在社交媒体如微博中,一条微博除了文本内容外,往往会带有用户的位置,可视为一个空间-文本对象.在很多移动应用中,用户的描述和位置可供查询,因而也是一个空间-文本对象.

一个典型的空间关键词查询(spatial keyword query,简称 SKQ)q 包括以下部分:查询位置 $q.loc$、查询词集合 $q.kws$、约束集合 $q.C$、排序公式 $q.f$ 和返回结果的数量 $q.k$.给定一个空间-文本对象集合 D 和空间关键词查询 q,空间关键词搜索从 D 中找出所有或者 $q.k$ 个空间-文本对象,使得每个对象 o 满足查询 $q.C$ 中的约束,并根据排序公式 $q.f$ 排列输出.

现有文献所研究的空间关键词查询并不完全相同,其主要区别在于查询约束和排序公式.

(1) 查询约束

由于对象和查询均包含空间和文本信息,查询约束一般包括空间位置约束和文本约束.其中,空间位置约束指定了 $o.loc$ 与 $q.loc$ 之间的关系,如相交[4−7]、包含或者被包含[4];而文本约束则指定了 $o.kws$ 与 $q.kws$ 之间的关系.可以分为 4 种.

- 完全匹配:要求 $o.kws \supseteq q.kws$,一般称这种语义为 AND 语义.
- 部分匹配:要求 $o.kws$ 与 $q.kws$ 部分匹配,即 OR 语义.对于这种约束,一般会在排序公式中考虑文本相关性.

- 模糊匹配:要求 *o.kws* 与 *q.kws* 模糊匹配.对这种约束,一般会指定字符串相似度计算公式(如编辑距离)和阈值.
- 布尔约束:用一个布尔约束来指定包含哪些关键词、可包含哪些关键词以及不包含哪些关键词.

(2) 排序公式

排序公式中,一般考虑了空间距离或/和文本相关性.例如,文献[8−12]仅根据查询结果离查询位置的距离排序;而有些文献同时考虑空间距离和文本相关性,如文献[7,13−16]采用的是距离和文本相关性的线性组合,文献[17]以距离和文本相关性的比值作为排序依据.

根据文献中 SKQ 的空间约束、文本约束和排序公式,可以将这些 SKQ 进行区分,见表 1.在表 1 中,排序公式中的线性组合指的是空间距离和文本相关性的线性组合,"−"表示没有指定该约束或者公式.

Table 1 Various types of spatial keyword query in the literature

表 1 文献中的各种空间关键词查询

序号	空间约束	文本约束	排序公式	文献
1	−	AND/OR	线性组合	[18,19]
2	−	AND	空间距离	[8,10,12]
3	−	OR	线性组合	[7,10,13,15,20,21]
4	−	布尔约束	空间距离	[9]
5	包含	OR	线性组合	[6]
6	相交、包含、被包含	AND	−	[4,22,23]
7	相交	AND	线性组合	[5]
8	−	模糊	线性组合	[24]
9	−	模糊	空间距离	[25,26]
10	包含	模糊	−	[25−27]

1.2 面临的挑战

空间关键词搜索同时涉及到空间查询和关键词的搜索,这给查询处理带来了极大的挑战.挑战主要在以下方面.

(1) 不同技术的集成.表面上看,SKQ 就是空间查询和关键词查询的集成,但是传统的空间查询和关键词搜索使用不同的索引技术和查询算法,如从索引技术来说,空间查询广泛使用 R-tree,Quadtree 等树结构,而关键词搜索使用倒排列表等较为扁平的结构,它们的集成面临不少困难.

(2) 有效的剪枝策略.由于搜索空间较大,查询处理的关键在于剪枝,根据空间距离剪枝或者根据文本相关性剪枝都会导致无用计算和大量无用的中间结果.如何构造更有效的剪枝策略是一个关键问题,也是难点.

(3) 灵活的排序公式.现有的空间关键词搜索技术大都基于少数几个预定义好的排序公式,能否支持任意的排序公式,是一个很大的挑战.

(4) 可伸缩性.关键词搜索和空间查询中都面临着可伸缩性的挑战.对于空间关键词搜索,这一挑战更加严峻.

2 空间关键词搜索技术

传统的空间查询和关键词搜索使用不同的索引技术和查询算法,为了有效地处理 SKQ,需要将空间索引和文本索引结合起来,形成空间-文本索引,并提出相应的查询算法.根据空间索引和文本索引结合方式的不同,可以将现有的查询处理技术大致分为 3 类:松散组合、空间优先、文本优先,如图 1 所示.

注意:分类中,R-tree 泛指 R-tree[28]或者 R*-tree[29].

在介绍这些类别之前,先给出一个例子,这个例子将贯穿本文始终.

图 2(a)显示了一个二维平面上的空间-文本对象集合和查询.在图中,圆点表示空间-文本对象,而查询则用星形表示.对象的文本信息如图 2(b)中所示,而建立在对象空间位置之上的空间索引(R-tree)如图 2(c)所示.

注意:在图 2(b)以及以下的例子中,对关键词信息进行了简化,省略了关键词频率等信息.

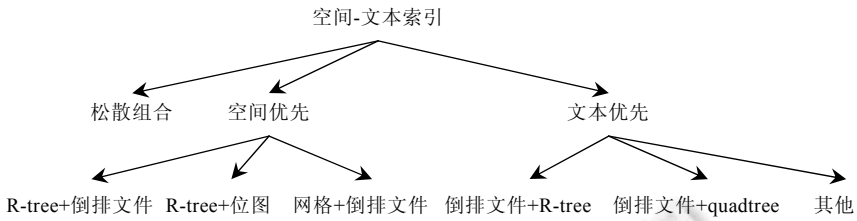
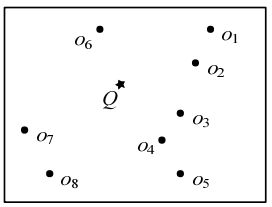


Fig.1 Classification of spatio-textual indices

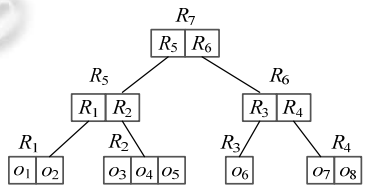
图 1 空间-文本索引分类



(a) 空间-文本对象

对象	关键词
o_1	平价牛排餐厅
o_2	牛排餐厅电影院
o_3	特色电影院
o_4	商场餐厅
o_5	商场电影院
o_6	平价电影院
o_7	特色餐厅
o_8	特色牛排餐厅商场

(b) 对象的关键词信息



(c) 空间索引

Fig.2 Spatio-Textual objects and query object

图 2 空间-文本对象和查询

下面将分别介绍每一种类型的查询处理技术.

2.1 松散组合

在这一类方案中,分别针对空间数据建立空间索引(一般采用 R-tree 或者 quadtree),针对文本数据建立文本索引(一般是倒排索引).两种类型的索引之间没有或只有松散的联系.查询处理时,分别从空间索引和文本索引出发,找出满足空间约束和文本约束的对象,然后将它们综合起来.

这类方案最典型的例子是文献[4].其中,位置信息用 R-tree 进行索引,而文本信息用倒排文件索引.R-tree 和倒排文件没有直接联系.查询处理时,分别在两种索引中查找候选对象,然后合并.

文献[30]中也采用 R-tree 和倒排文件分别索引空间信息和文本.但是,R-tree 中的每个节点都添加了一个标记(label),表示了从根到该节点的路径.与此同时,倒排文件中的每个对象也带了标记.该标记可以在不访问 R-tree 的情况下判断一些空间关系.查询处理时,首先从倒排文件中检出查询词的倒排列表,将它们进行合并;然后,根据合并的结果构造虚拟的 bR-tree[31].在这个过程中,对象的标记将会发生作用.然后,利用 bR-tree 的查询算法来进行查询.

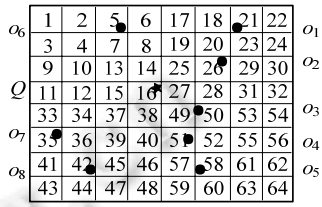
SFC-QUAD[6]索引首先根据对象的空间位置构造空间填充曲线(space filling curve)编码给对象赋以 ID.索引包括两个部分:quadtree 和倒排文件.quadtree 用于索引对象的空间位置,倒排文件用于索引对象的文本,其中用到了对象的空间填充曲线 ID.此外,在 quadtree 和倒排文件之间并没有联系.查询处理时,首先在 quadtree 中找出若干空间范围,根据该范围获取到若干对象 ID 范围;然后,根据对象 ID 在倒排文件中的若干块中进行查找.这里利用了空间填充曲线的一个特点:空间相邻的对象其空间填充曲线编码也相邻.在该算法中,空间索引可以用于减少倒排文件的扫描范围.

文献[20]中则仅使用了倒排列表一种数据结构.对每个关键词,维护了两个倒排列表:文本倒排列表和空间倒排列表.其中,文本倒排列表存放了包含该关键词的对象(文档)标识 docID 和关键词得分(权重),按照得分降序排列;空间倒排列表存放了包含该关键词的 docID 和对象的 Z-order,按照 Z-order 顺序排列.为了减少随机 IO,

传统信息检索系统中的全局对象(文档)列表被分裂成了多个.每个倒排列表有一个对象列表与之对应,包含了这个倒排列表中出现的对象的信息.针对本文所示例子建立的索引结构如图 3 所示,其中,文本倒排列表的对象根据简化的 tf.idf 得分排列(具体得分没有显示出来);空间倒排列表中列出的是对象的 Z-order.对象与 Z-order 之间的关系在图 3(b)中显示出来(对象标识符在左侧或者右侧显示).

关键词	文本倒排列表	空间倒排列表
平价	o_6, o_1	5, 21
牛排	o_1, o_2, o_8	21, 26, 42
餐厅	o_4, o_7, o_1, o_2, o_8	21, 26, 35, 42, 51
电影院	o_3, o_5, o_6, o_2	5, 26, 50, 58
特色	o_3, o_7, o_8	36, 42, 50
商场	o_4, o_5, o_8	42, 51, 58

(a) 文本倒排列表和空间倒排列表



(b) 对象的 Z-order

Fig.3 Inverted lists of spatio-textual objects

图 3 空间-文本对象的倒排列表

文献[20]提出了基于 top-k 算法 CA[32]的查询算法.算法逐个列表扫描,一个列表一次扫描的范围依赖于一个得分区间.对每个列表,在扫描过程中,更新所遇到的每个对象的估算分,并维护得分最高的 k 个对象 $TopK$ 以及这 k 个对象的最小值 W_k.如果一个对象的上界大于 W_k,则称该对象为可行的(viable).每一次扫描后,随机访问每个可行对象在所有列表中的分数,并更新 W_k 以及其他未见到对象的上界 B_k.如果 B_k 小于 W_k,则停止扫描,计算出所有可行对象的准确得分,以及最终的 top-k 结果.

2.2 空间优先

在这类查询处理技术中,空间-文本索引是对空间索引进行增强、加入文本信息得到的.空间索引主要是使用 R-tree,也有少数工作使用网格索引.

由于这类技术有很多共同之处,下面首先介绍索引技术,然后介绍查询算法.

2.2.1 索引技术

1) R-tree+倒排文件

这种方案利用 R-tree(或者 R*-tree,以下统称为 R-tree)来索引位置信息,同时,在 R-tree 中的某些或者所有节点处建立倒排文件来索引这些节点所包含的文本.

(1) R*-IF 索引[4]

在 R*-IF 索引中,R-tree 将所有对象的地理位置索引起来,对于 R-tree 中的每个叶节点,创建一个倒排文件,索引了被该叶节点所包含的对象的文本.例如,在图 2 所示的对象集合上建立 R*-IF 索引后,其中的两个叶节点所关联的倒排文件如图 4 所示.

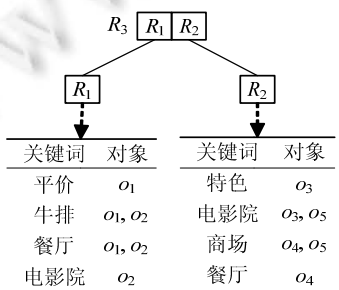


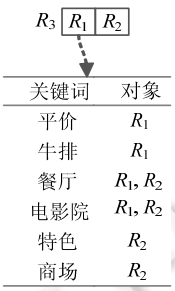
Fig.4 Examples of R*-IF index

图 4 R*-IF 索引示意图

(2) IR-tree 及其变体[13,21]

IR-tree 给 R-tree 中的每个节点关联了一个倒排文件,倒排文件记录了每个关键词在哪些子节点中出现.例如在图 5 中,中间节点 R_3 所关联的倒排列表记录了关键词在子节点 R_1 和 R_2 中的出现信息.

IR-tree 还有若干变体,包括 DIR-tree,CIR-tree 和 CDIR-tree.DIR-tree 在构造 R-tree 的时候,同时考虑了空间和文本信息,使得一个节点包含的对象在空间上尽可能地接近,而在文本上尽可能地相似.CIR-tree 根据文本描述将对象聚集到若干簇中,然后在每个节点处记录了关键词在各个簇中的分布情况.例如,对于图 2 所示的对象集合,假定文档属于 3 个集合:C_1(含 o_1,o_2,o_8),C_2(含 o_3,o_5,o_6)和 C_3(含 o_4,o_7),那么节点 R_3 的倒排文件中包含的信息如图 6 所示,其中记录了每个关键词在哪个子节点的哪个簇中出现.CDIR-tree 则将 DIR-tree 和 CIR-tree 结合起来.即,首先构建 DIR-tree,然后在此基础上增加对象的聚类信息.



关键词	对象
平价	R_1
牛排	R_1
餐厅	R_1, R_2
电影院	R_1, R_2
特色	R_2
商场	R_2

Fig.5 Inverted lists of internal nodes

图 5 中间节点的倒排列表

关键词	对象
平价	$R_1.C_1$
牛排	$R_1.C_1$
餐厅	$R_1.C_1$, $R_2.C_2$, $R_2.C_3$
电影院	$R_1.C_1$, $R_2.C_2$
特色	$R_2.C_2$
商场	$R_2.C_2$, $R_2.C_3$

Fig.6 Inverted lists in CIR-tree

图 6 CIR-tree 中的倒排列表

(3) WIR-tree 索引[12]

WIR-tree(W 代表 word partitioning)是 IR-tree[13,21]的变体,它与 IR-tree 的区别是对象聚集的方式不同.WIR-tree 根据包含的关键词情况将对象分组,使得每个组共享的关键词尽可能地少.

以图 2 所示的对象集合为例.首先,根据关键词的出现频率对它们排序,假定顺序为:餐厅、电影院、商场、特色、牛排、平价.根据这个顺序,将所有的对象划分到若干个组,划分过程可以用图 7 来描述.每个节点对应了一个对象集合.在每一步,根据是否包含某一关键词(用+和–表示包含或者不包含)将当前节点一分为二.例如,首先根据是否包含“餐厅”,将原始对象集合 D 分为 D_1 和 D_2,然后根据是否包含“电影院”进一步划分.当产生的对象集合的大小不超过指定阈值(如 2)时,该集合不再划分.例如,图 7 中最后产生 5 个对象集合 D_3,D_7,D_8,D_5 和 D_6.如果所有关键词都已经用完,还不能将当前对象集合划分成足够小的集合,则在当前对象集合上构建 R-tree,并将 R-tree 的叶节点作为划分的结果.

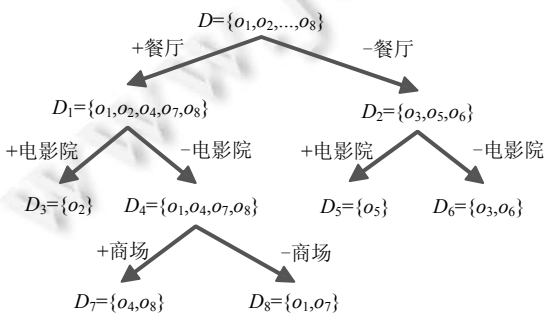


Fig.7 Process of partitioning objects in WIR-tree

图 7 WIR-tree 中对象分组的过程

然后,在划分产生的对象集合基础上构造 WIR-tree.构造的过程是自底向上的:首先,上一步产生的对象集合作为叶节点;然后,将每个叶节点视为一个对象,重复上述对象分组的过程,产生上一层节点.这一过程一直继续,

直到某一层只有 1 个节点为止.这样产生的树就是 WIR-tree.与 IR-tree 类似,WIR-tree 也在每个节点处保留了一个倒排文件,记录了关键词在子节点中的出现情况.

(4) IR-tree(Li)索引[7]

IR-tree(Li)和 IR-tree[13,21]的思想类似,其区别在于,IR-tree(Li)仅在每个叶节点处关联了倒排文件,每个非叶节点 i 包含了一个概要 $\left\langle A_i, |D_i|, \bigcup_{w\in W_i}\{df_{w,D_i}, TF_{w,D_i}\}\right\rangle$,其中,$A_i$ 是节点 i 所包含的对象的 MBR(minimum bounding rectangle,最小包围矩形),$|D_i|$是节点 i 所包含的对象的个数,$\bigcup_{w\in W_i}\{df_{w,D_i}, TF_{w,D_i}\}$ 则记录了在节点 i 的对象中每个关键词的文档频率、关键词频率等信息.

(5) KR*-tree (keyword R*-tree)索引[23]

KR*-tree 索引中,R*树的每个节点记录了所包含的对象的关键词集合.此外,对每个关键词,以倒排列表的方式记录了在哪些节点中出现.对于本文图 2 中的例子,关键词列表如图 8 所示.

关键词	节点列表
平价	R_1, R_5, R_7, R_3, R_6
牛排	R_1, R_5, R_7, R_4, R_6
餐厅	R_1, R_5, R_7, R_2, R_4, R_6
电影院	R_1, R_5, R_7, R_2, R_3, R_6
特色	R_2, R_5, R_7, R_4, R_6
商场	R_2, R_5, R_7, R_4, R_6

Fig.8 Keyword lists in KR*-tree

图 8 KR*-tree 中关键词列表

(6) LBAK-tree[27]

与 IR-tree 类似,LBAK-tree 也是在 R-tree 索引基础上增强了倒排文件.但是 LBAK-tree 支持关键词的模糊匹配,因此与 IR-tree 有所区别:LBAK-tree 中的倒排文件存储的是 q-gram 的倒排列表,而且并非所有中间节点都关联了倒排文件,而是精心选择的某些中间节点,使得索引大小和查询代价之间达到平衡.

2) R-tree+基于位图的索引

另一种常用的文本索引是基于位图(bitmap-based)的索引,主要包括签名文件(signature file)和位图索引(bitmap index).因此,也有文献提出将 R-tree 与基于位图的索引结合起来,形成空间-文本索引.

(1) IR2-tree 索引[10]

在这个索引中,R-tree 中的每个节点关联了签名文件,一个签名文件实质上是一个位图,概括了一个节点所包含的关键词信息.具体而言,叶节点中对应于每个对象包含了一个签名文件,记录了该对象中的关键词信息.在一个对象的签名文件中,如果对象包含了某关键词,则通过哈希函数找出若干位置,将这些位置上的比特置 1.在中间节点中,对应于每个子节点也包含了一个签名文件,它是通过将下一层的签名文件叠加得到的.签名文件是直接存储在节点中的.

图 9 是图 2 对象集合上构建的 IR2-tree 索引.其中,为了方便理解,签名文件被简化:每个比特对应于一个关键词,依次为餐厅、电影院、商场、特色、牛排、平价.1 表示节点包含了对应的关键词,0 表示不包含.

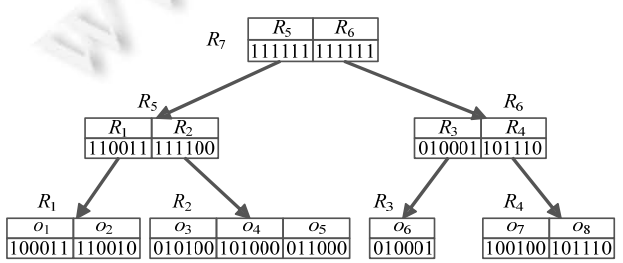


Fig.9 An example of IR2-tree

图 9 IR2-tree 实例

(2) SKI 索引[9]

图 10 是一个 SKI 索引的示意图.一个 SKI 节点分为两个部分:空间索引部分和文本索引部分.空间索引部分基于 R-tree,其中,具有相同父节点的叶节点的集合称为超节点,如 S_1,S_2.在叶节点的每个条目(entry)中,记录了自己所在的超节点;在中间节点的每个条目中,记录了子树中超节点的范围.文本索引部分则以位图的形式记录了关键词在各个对象中的出现情况,按超节点排列.例如"平价"对应的位图为 10000100,表示该关键词在 o_1,o_6 中出现.

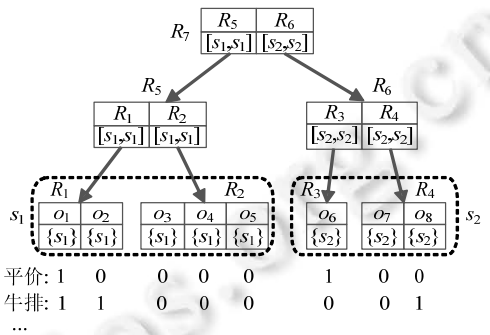


Fig.10 An illustration of SKI index

图 10 SKI 索引示意图

(3) bR-tree 索引[31]

bR-tree 与 IR^2-tree 有类似之处:bR-tree 中,R-tree 的每个节点带有一个位图,总结了该节点所包含的关键词的信息.与此同时,每个节点除了记录了节点的 MBR 之外,还记录了关键词的 MBR.关键词的 MBR 表示了该节点下该关键词的空间分布范围.为了节省空间,每个关键词 MBR 采用了一种紧凑的近似表示.

(4) MHR-tree[25]

为了支持空间-文本对象上的模糊字符串搜索,文献[25]提出了 MHR-tree 结构.在这种结构中,索引中间节点维护了所包含的关键词的 q-gram 的 min-wise 签名,该信息可以用于剪枝未包含相似字符串的节点.

3) 网格+倒排文件

文献[33]研究了无线数据广播环境下的空间关键字查询问题,提出了一种索引 IRGI,该索引为 2 层索引:第 1 层为网格索引,包括网格划分密度以及每个网格的下一次广播时间;第 2 层为每个网格的倒排列表索引.

2.2.2 查询算法

空间优先方案主要是以 R-tree 为基础,将 R-tree 与文本索引结合起来,实现快速过滤.大部分的算法可以用图 11 给出的算法框架来描述.

该算法是根据著名的 best-first 遍历算法[34]得到的.该算法使用一个优先队列 *Queue* 来保存需要考察的节点和对象,Top-K 用来保存已产生的结果.在队列中,以节点或者对象相对于查询的距离为键进行排列.算法首先将根节点放入队列中.每次从优先队列中取出当前最好的元素(即,队头的元素)n.如果 n 是一个对象,且 n 到查询的距离小于队头元素到查询的距离,则将 n 加入到结果集中.如果 n 是一个叶节点,则将 n 包含的每个对象及其距离加入到队列中;如果 n 是一个中间节点,则将 n 的每个条目及其估算距离加入到队列中.如此展开,直到产生了 $q.k$ 个结果为止.注意:对象到查询的距离是直接计算的,而节点到查询的距离则是估算的.距离如何定义以及如何估算距离,在不同文献中也是有所不同的.不失一般性,假定一个对象 o 到查询 q 的距离定义为

$$d(o,q)=\alpha\cdot dist(o.loc,q.loc)+(1-\alpha)\cdot(1-sim(o.kws,q.kws)).$$

即对象 o 到 q 的距离是 o 和 q 的空间距离和文本距离的线性组合.

当遍历到节点 n 时,需要计算 n 所包含的节点到 q 的距离的最小值,即 $\min_{o\in n}d(o,q)$,这一值一般根据节点 n 所拥有的信息进行估算,即

$$\min d(n,q)=\min_{o\in n}d(o,q)\approx\alpha\cdot dist(n.rect,q.loc)+(1-\alpha)\cdot(1-sim(n.kws,q.kws)),$$

其中,*n.rect* 表示 *n* 对应的矩形,*n.kws* 表示 *n* 所包含的关键词,*dist*(*n.rect*,*q.loc*)表示节点 *n* 对应的矩形与查询点之间的空间距离.对于不同的索引,在每个节点处所保留的关键词信息有所不同,因此,计算 *sim*(*n.kws*,*q.kws*)时有所不同.

算法 1. 基于 R-tree 的空间关键词查询算法框架.
输入:查询 *q*、索引 *I*.
输出:*q.k* 个查询结果.

```
1:  Queue←NewPriorityQueue()
2:  TopK←∅
3:  Queue.push(I.root,0)
4:  While Queue≠∅do
5:    (n,d)←Queue.pop()
6:    If n 是对象 then
7:      If Queue≠∅ and d>Queue.top().key then
8:        Queue.push(n,d)
9:      Else
10:       TopK.insert(n)
11:   If |TopK|=q.k
12:   break
13:     Else if n 是叶节点 then
14:       For each entry(object) e in n do
15:         Queue.push(e,d(q,n))
16:     Else
17:       For each entry(node) e in n do
18:         Queue.push(e,mind(q,e))
19:   return Top-K
```

Fig.11 Algorithm framework of R-tree-based SKQ processing

图 11 基于 R-tree 的空间关键词查询处理框架

2.3 文本优先

这种方案中,空间-文本索引是对文本索引(如倒排文件)进行增强,将每一个关键词映射到一个含有空间信息的数据结构上.下面对几种典型的技术分别介绍索引技术和查询算法.

2.3.1 倒排文件+R-tree

(1) IF-R*索引[4]

在 IF-R*索引中,对每个关键词建立一个单独的 R-tree,索引了包含该关键词的对象的地理位置,如图 12 所示.查询时,首先根据关键词找到对应的 R-tree,然后合并产生最终结果.

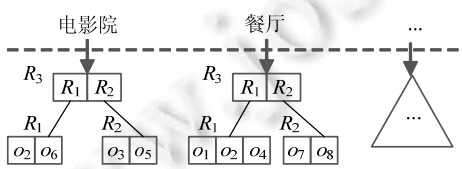


Fig.12 IF-R* index structure

图 12 IF-R*索引结构

(2) S2I 索引[15]

S2I 的基本结构是倒排文件,但是频繁关键词和非频繁关键词对应的文件是不同的:对于频繁关键词,将所有包含该关键词的对象保存在一个 aR-tree(aggregate R-tree)中,其中一棵 aR-tree 在 R-tree 的基础之上还在每个节点处记录了该关键词的最大相关性;而对于非频繁关键词,与其有关的对象信息保存在一个块文件(block file)中.

S2I 的查询处理算法分为单关键词算法和多关键词算法.单关键词算法只需要对一个文件或者 aR-tree 进行检索,比较简单.下面介绍多关键词算法.

由于 S2I 将每个关键词分别映射到一个文件或者 aR-tree,在查询时,首先可以获取每个关键词对应的数据结构(称为源).为了提高合并效率,该算法将计分公式进行了分解.

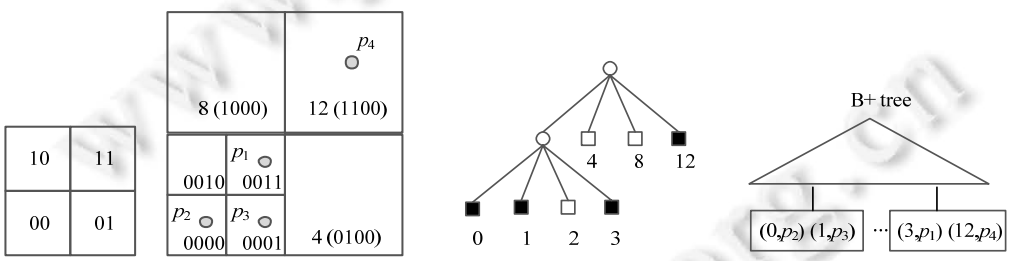
对于距离公式 $d(o,q)$,将其分解为 $d(o,q)=\sum_{t\in q.kws} d_t(o,q)$,即若干部分距离的叠加,每个部分距离都是从一个关键词计算得到的.对于 $d_t(o,q)$,根据计分公式推导出一个上界和下界.算法按照得分降序从每个源中获取下一个对象.每次从关键词 t 的源 S_t(块文件或者 aR-tree)中获取一个对象 o,首先计算 $d_t(o,q)$,然后更新 o 相对于 q 的得分下界以及 S_t 中其他对象得分的上下界,根据这些信息,可以估算 o 的得分范围,并尽早剪枝.

2.3.2 倒排文件+quadtree

(1) IL-Quadtree 索引[8]

IL-Quadtree(inverted linear quadtree)索引使用 quadtree 对整个空间进行划分,并根据空间填充曲线编码 Morton 编码对节点进行编码.Quadtree 每次将一个区域划分成为 4 个子区域,可以分别用 00,01,10 和 11 进行编码,这样,将每一次划分的编码连接起来,就可以得到一个区域的 Morton 编码.例如在图 13(a)中,对于 p_1 而言,第 1 次划分时所在区域为 00,第 2 次划分时所在区域为 11,因此,p_1 所在区域的 Morton 码为 0011.p_4 第 1 次划分时所在区域为 11,第 2 次没有划分,但是仍然补齐,使得所有区域的编码长度相同,因此,p_4 所在区域为 1100.对于所有非空的叶节点,再建立一棵一维的 B+-tree 来对它们进行索引,该索引按照 Morton 编码有序.

图 13(b)是用树状结构描述的 quadtree,其中,节点的编码用整数而非二进制串表示.方块表示叶节点,而圆形表示中间节点.黑色方块表示的是非空区域,而白色方块则表示空区域.IL-Quadtree 则是倒排文件和线性 quadtree 的结合.对于每个关键词,维护一个线性 quadtree,用于索引所有包含了该关键词的对象,如图 13(c)所示.



(a) Morton 编码 (b) quadtree 树状结构 (c) 线性 quadtree

Fig.13 IL-Quadtree index structure

图 13 IL-Quadtree 索引结构

查询处理算法如下:与 best-first 算法类似,算法也使用一个堆结构来保存当前遇到的最好的若干个节点,每次从中选择一个最好的节点进行展开.堆结构的初始内容是各查询词的 quadtree 的根节点.算法的特别之处在于:如果 e 是一个黑色叶节点,则检查 e 所覆盖区域与其他 quadtree 中的黑色叶节点是否重叠,如果没有重叠,那么 e 不可能包含所有关键词,因而 e 会被忽略.通过 IL-Quadtree 这种数据结构,检查两个区域是否有重叠非常容易,从而使得合并多个列表中对象变得容易,从而极大地提高了查询的效率.

(2) I^3 索引[18]

该索引将位置空间用 quadtree 进行划分.由于不同的关键词在一个单元格中出现次数不同,因此将(关键词,单元格)对作为考虑的基本单位,称为关键词单元格(keyword cell).很明显,不同的关键词单元格中包含的对象数是不同的.如果一个关键词单元格中的对象数超过阈值,则称其为稠密的(dense).该阈值取决于一个磁盘页中能够容纳的对象数量.

对于频繁或者非频繁的关键词而言,在索引中处理方式是不同的:对于非频繁的关键词,一个页面就足够存储相关对象,此时,通过查找表直接将其映射到数据文件中的物理位置;对于频繁的关键词,需要用到多个磁盘页,为了更灵活地存取这些页面,该索引设计了头文件.头文件是一个类似于 quadtree 的结构,其中每个节点存储

了关于该关键词的概要信息,包括签名和最大文本相关性.如果一个节点是稠密的,则将该节点进行划分,因此,头文件中的每个叶节点都是非稠密的.头文件中的叶节点含有指针,指向该关键词单元格的磁盘页.

因此,I^3 索引由 3 部分来构成:查找表、头文件和数据文件,如图 14 所示.数据文件存放了所有关键词的倒排列表,它由若干磁盘页组成,每个磁盘页对应于一个非稠密关键词单元格.头文件包含了频繁关键词的 quadtree 的概要信息.查找表将一个关键词映射到头文件或者数据文件中.如果关键词是非频繁的,则从查找表中可以直接查询到数据文件中对应的磁盘页;否则,可以查找到其头文件的起始位置.

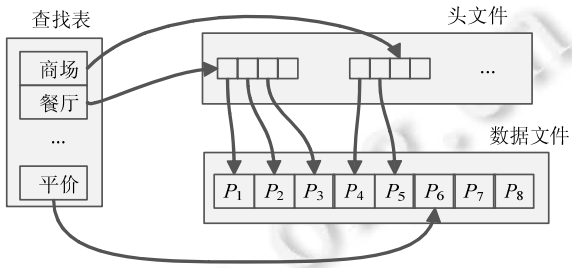


Fig.14 I^3 index structure

图 14 I^3 索引结构

在查询时:对于非频繁关键词,直接读取对应的磁盘页;对于频繁关键词,查询算法在某种程度上类似于 best-first 算法.算法也使用一个优先队列,其中的元素是 quadtree 的节点,并按照节点的估算相关性排序,每个节点还记录了当前的稠密关键词.初始时,整个初始空间压入队列中.算法每次从队列中取出队头节点 C.如果 C 中没有稠密关键词,则直接读取对应的磁盘页,并进行合并;否则,将 C 的子节点压入队列中,估算每个子节点的得分,并更新子节点的稠密关键词.

2.3.3 其 他

SFC-SKIP[6]方案在普通的倒排索引的基础上加入了一些空间分布信息.在 SFC-SKIP 中,每个倒排列表由若干个块构成,为了能够根据空间约束快速地在块之间跳转,在块上面加入了若干层的空间信息.在文献[6]中:在第 1 层中,对每 4 个块将存储一个 MBR;在第 2 层,每 16 个块存储一个 MBR;而在第 3 层,每 64 个块存储一个 MBR,如图 15 所示.在查询时,首先根据各个层上存放的 MBR 信息缩小查询范围,然后读取相应的块进行验证.

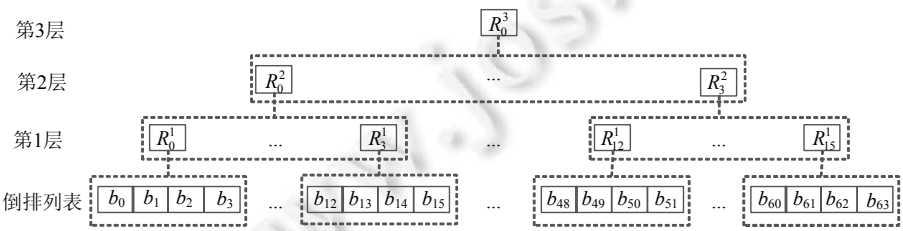


Fig.15 Illustration of SFC-SKIP

图 15 SFC-SKIP 示意图

2.4 比 较

本节对已有的关键词查询处理技术进行分析,比较从以下方面展开:

- 查询类型:查询处理技术应该支持尽可能多的查询类型.
- 查询效率:这是最重要的考虑因素,它主要取决于索引的剪枝能力.
- 索引代价:包括构建索引的时间和索引大小.
- 可伸缩性:当查询词数量和数据量大小变化时的查询性能.

- 兼容性:查询处理技术应尽可能地复用系统中已有的组件.

上述 3 类查询处理技术各有优缺点,其比较见表 2.

Table 2 Comparison of query processing techniques

表 2 各种查询处理技术的比较

性能因素	松散组合	空间优先	文本优先
查询类型	较多	多	较多
查询效率	简单查询效率好 复杂查询有待验证	复杂查询效率较好	复杂查询效率较好 特定类型查询效率很好
索引代价	低	较低	高
可伸缩性	好	好	一般
兼容性	兼容空间索引和文本索引	兼容空间索引	较差

(1) 松散组合方案

对于简单的空间关键词查询,如不涉及到排序的查询,这种方案的效率很好[35];但是对于复杂的查询,由于空间索引和文本索引之间的耦合较为松散,很难实现复杂的过滤,因而剪枝能力较弱.但是最近,文献[20]声称,其提出的技术比另外两种方案都要优秀,因而这一类别的技术还有深入研究的空间.这一方案基于现有的非常成熟的索引技术,因而索引代价较低,可伸缩性较好;不需要对现有的空间索引和文本索引做太多的改造,实现简单,兼容性好.

(2) 空间优先方案

这种方案能够处理各种类型的空间关键词查询.它在已有的空间索引的基础上进行增强,但需要注意的是:索引中的文本信息并不是与空间索引存在一起,因而不需要对已有的空间索引做太多的改动,因此特别适合于已有的基于位置的应用.查询算法也可以兼容已有的空间查询算法,如最近邻查询算法,但是性能不一定最好[32].索引代价较低,可伸缩性较好.

(3) 文本优先方案

这种方案也能支持较多的查询类型.索引沿用了倒排文件的思想,但为了有效地剪枝,倒排文件往往进行了精心设计,如 S2I,IL-Quadtree 以及 I^3 等,这些结构是对传统倒排索引的大幅改造,对于已有的应用势必带来影响.从实验性能来看,当查询词较少时,这类方案往往能够提供最好的性能.其缺点是,其性能对查询词数量和文本长度较为敏感.另外,这种方案的索引时间和索引大小往往是最大的[35],这也在一定程度上影响了其实用性.

3 空间关键词搜索的扩展

随着研究的深入,研究者发现有些新问题不在前面所述问题框架之内,于是提出了一系列新问题,对现有研究进行了扩展,主要的扩展如下:(1) 从欧式空间扩展到路网模型;(2) 从静态空间关键词搜索到移动空间关键词搜索;(3) 从空间关键词搜索到时空关键词搜索.下面分别介绍这 3 个方面的工作.

3.1 路网中的空间关键词搜索

在很多实际的问题中,对象和查询均处于路网(road network)中.路网模型无疑使问题变得更加复杂:首先,在建立空间索引时,不仅要索引对象的空间属性,还要索引路网信息以及对象所在的路段信息;其次,仅根据空间属性无法计算距离,还需要结合路网信息,使得对象之间距离的计算和估算更加复杂.

目前,仅有文献[16]研究了路网中的空间关键词搜索问题.文中提出了由 4 个部件构成的索引结构,如图 16 所示.这 4 个部分的构成和作用分别如下:

(1) 空间组件存储了路段的信息.每个路段是一条折线,用一种称为网络 R-tree 的索引结构管理路段的 MBR,路段的具体信息,如起点、终点等则存储于折线文件中.这一组件可以用于定位查询点所在的路段.

(2) 邻接组件存储了节点之间的邻接信息,可用于查询一个节点的邻接节点.给定一个节点 v,邻接 R-tree 指向了一个块,其中存放了 v 的所有邻接节点,并记录了边的 *id* 和长度.

(3) 给定一个边的 *id* 和关键词 *t*,映射组件将它们映射到一个倒排列表,并记录了 *t* 在这条边上的对象中的最大权重.倒排列表组件中包含了倒排列表信息和词汇表,对于每个对象 *p*,倒排列表中记录了 *p* 到边的一个端点的距离以及关键词 *t* 在这个对象中的权重;而词汇表则包含了每个关键词的文档频率 *df* 等信息.

查询过程如下:首先定位到查询点所在的边;然后,沿着边进行扩展,在此过程中,计算候选对象与查询点之间的距离,并将候选对象存入一个堆结构中.这一过程继续,直到不可能产生分数更高的对象为止.

为了进一步提高查询性能,文献[16]中还提出在路网上构建覆盖网络(overlay network).覆盖网络由一些区域构成,每个区域对应了路网中的一个范围,并维护了一个伪文档,概括了所覆盖的对象的文本信息.区域之间有边界点,它们之间存在虚拟边相连.通过覆盖网络,可以直接跳过不相关或者相关性低的区域,从而提高剪枝能力.

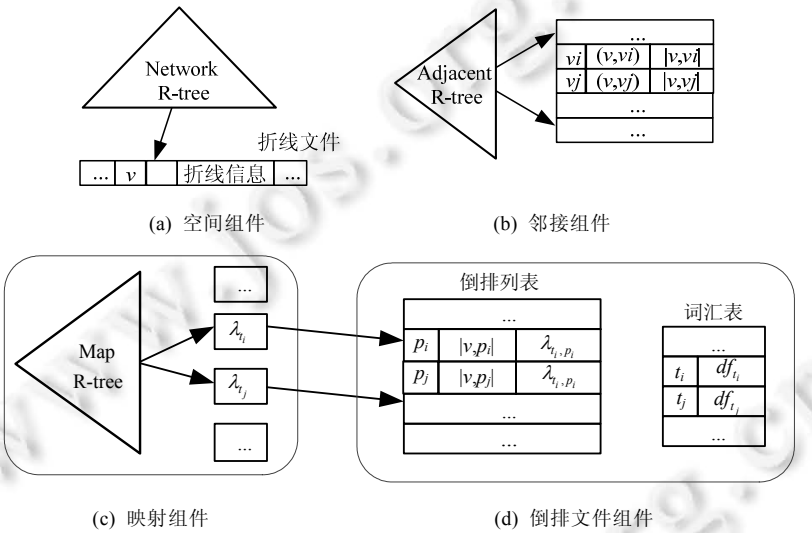


Fig.16 Spatio-Textual index on road network

图 16 路网上的空间-文本索引

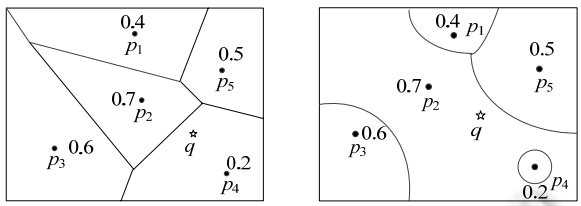
3.2 连续空间关键词搜索

在移动应用中,查询者在持续移动,要求能够持续地返回查询结果,这一问题称为连续(移动)空间关键词搜索(continuous spatial keyword search,简称 CSK).由于移动一般基于实际道路,因此往往在路网空间中考虑这一问题.除了在第 1.2 节介绍的那些挑战外,CSK 问题还面临一个额外的挑战,那就是如何持续地返回最新的结果.

移动空间查询的典型处理方法是查询感知处理法.该方法根据连续查询的查询条件计算出安全区(safe zone)[36,37],只有当移动对象离开或进入安全区时才更新查询结果,而 Voronoi 图经常用来计算安全区.Voronoi 图的一个简单例子如图 17 所示,其中,$p_1,\ldots,p_5$ 是空间对象.Voronoi 图将空间划分为 5 个单元格,每个对象属于一个单元格.Voronoi 单元格有一个特征,即一个对象所在的单元格中的所有点都以该对象为最近邻,因而一个单元格可以作为一个对象的安全区.例如,假定 q 为查询点,在 p_4 的单元格中,那么只要查询 q 没有离开该单元格,p_4 都是 q 的最近邻.

但是,Voronoi 图没有考虑文本相关性,因而无法适用于空间关键词查询.为了使用 Voronoi 图,文献[17]将查询和对象之间的距离定义为空间距离和文本相关性的比值,然后使用 MW-Voronoi(multiplicatively weighted Voronoi)图来描述安全区.一个对象的 MW-Voronoi 单元格根据对象的权重定义.类似地,在 MW-Voronoi 图中,一个对象的安全区就是它的 MW-Voronoi 单元格.也就是说,该单元格中的所有点到该对象的加权距离都比到其他对象的加权距离小.图 17(b)显示了一个 MW-Voronoi 图,其中每个对象有个权重,即对象和查询的文本相关

度.在该图中,查询 q 处于对象 p_2 的安全区中,因此距离 q 最近的对象是 p_2.



(a) Voronoi 图 (b) MW-Voronoi 图

Fig.17 Voronoi and MW-Voronoi diagram

图 17 Voronoi 和 MW-Voronoi 图

由于对象的文本相关性是查询相关的,因此不同查询时的 MW-Voronoi 图是不同的.也就是说,MW-Voronoi 图是需要在线计算的.文献[17]提出了两种计算安全区的算法 IBD 和 MSK-uvr,它们基于 IR-tree 索引,具有不同的剪枝能力.此外,文献还提出算法利用前一个安全区和结果的信息来计算下一个安全区.为了减少通信代价,文献提出计算保守安全区来代替实际安全区.

文献[14]针对一个更一般的排序公式,即距离和文本相关性的线性组合,研究了构建安全区的方法.在该文献中,一个对象的安全区是该对象的优势区域(dominant region)的交.给定两个对象 o^*和 o,o^*相对于 o 的优势区域是一个区域 R,使得对于 R 中的任意查询,o^*比 o 要好.优势区域具有一定的几何意义,因此安全区也具有几何意义,但是不同于 Voronoi 图,而且具有不同的形状.

文献[38]在路网模型中研究了移动空间关键词查询.类似于安全区,文献[38]提出了安全路段的概念.所谓安全路段,是一条边的一部分,使得查询者在该路段上移动时,top-k 查询结果保持不变.路段上的 top-k 查询结果有一个重要性质:假定查询位于边(n_1,n_2)上,那么查询的 top-k 结果要么是位于边(n_1,n_2)上的相关对象,要么是两个端点 n_1 或 n_2 的 top-k 结果.这一性质使得可以将动态的 top-k 结果计算转化为静态的网络节点遍历.文中提出了两个算法 QCA 和 OCA.两种算法都是首先计算查询结果以及安全路段.当查询结果需要更新的时候,两种算法以递增的方式来遍历路网,增量地更新查询结果,但是遍历网络的方式不同:QCA 每次从查询所在边的一个端点出发遍历,直至找到top-k结果,与此同时,它维护了一个扩展树(expansion tree)来避免随后的重复扫描;OCA 则从一个查询相关的对象出发遍历,它动态地构造一个 k 阶最短路径树(k-order shortest path tree),有了 k 阶最短路径树,只需要遍历很少的网络边就可以计算出 top-k 结果.

文献[39]研究了路网中空间关键字连续 k 近邻查询问题,提出了包含一棵 PMR-quad 树和 3 个内存表的数据结构以存储和索引路网结构信息、对象的位置和关键字信息,提出了可调节的综合距离值计算公式.为了实现查询的连续处理,通过监控有关候选对象的综合距离值的变化来对查询结果进行修正,以保证查询结果的持续有效性.

3.3 时空关键词搜索

随着多源信息的融合,越来越多的对象具有时间、空间和文本的多重属性,最典型的是社会媒体,如微博.一条微博除了其文本内容外,还会记录发布的时间以及发布的位置.因此,对于这样的数据,可以从时间、空间和文本这 3 个角度进行查询,也即时空关键词搜索.

文献[40]研究了时空关键词查询,查找在指定的时空范围中出现最频繁的关键词,例如在指定时间段和区域中发表的微博里出现次数最多的 k 个关键词.为此,文献建立了一个多层次的时空网格索引,它将时空空间划分为多个网格,网格具有不同的粒度.每个网格有一个预计算的摘要,即最频繁的词项及其计数.为了适应数据的变化,网格的摘要大小(计数器的数量)可以动态调整.

文献[41]研究了带时空约束的关键词查询,即在给定的时空范围内与关键词查询相关的对象.文中首先将

根据对象空间属性建立一棵类似于 R-tree 的树,树的每个叶节点关联了一个倒排索引,记录了所包含的对象的文本信息.为了表示时间属性,将对象的时间映射到对象标识符,这样,时间范围约束就可以映射为一个对象标识符的范围,可以在倒排索引中直接搜索.

文献[42]研究了时空关键词订阅(tempora1 spatial-keyword top-k subscription,简称 TaSK)问题.给定一个空间-文本对象流,其中每个对象有一个时间属性,一个 TaSK 查询带有一个查询位置和一个关键词集合,返回在空间、文本与时间上相近的 top-k 个空间-文本对象,其中,对象的相关性同时考虑了空间距离、文本相关性和时间新近性.处理 TsSK 问题的典型思路是:将某些查询归并到一组,使得可以对一组查询同时处理,从而提高处理效率.为了能够将 TaSK 查询归并到组,提出了条件影响域(conditional influence region,简称 CIR)的概念.对于一个普通的空间查询,其影响域是以该查询为中心、查询到第 k 个最近邻的距离为半径的区域,落在影响域中的对象会影响该查询的订阅结果.CIR 就是有条件的影响域,这里的条件就是文本相关性和时间.给定一个 TaSK 查询 q,文本相关性 tr 和时间戳 t,用 $CI_q(tr,t)$表示 q 的 CIR,如图 18 所示.

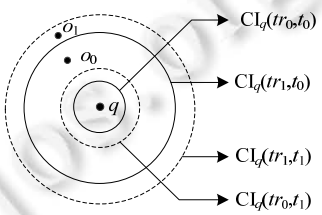


Fig.18 ClRs of query q

图 18 查询 q 的 CIR

根据 CIR,可以推导出针对对象的过滤条件.给定一个 TaSK 查询 q 和一个空间网格 c 以及当前 q 的第 k 个最近邻,可以算出一个对象有可能成为 q 的查询结果所应该具有的最小文本相关性.多个 TaSK 查询被索引起来,其中,索引根据查询的最小文本相关性分组,使得对一个组进行整体过滤成为可能.

4 其他相关工作

除了对空间关键词搜索进行了深入研究以外,研究者还研究了相关的其他问题:

- 面向动态关键词的 SKQ

文献[43]考虑动态关键词的情形,即对象的关键词可能是动态变化的,提出了一种支持动态关键词的空间-关键词混合索引树(DKR-tree).与 IR^2-tree 这样只能表示静态关键词与空间位置关系的索引树相比,DKR-Tree 将更新代价转化到了建立索引上,从而大大减少了在线更新代价,从整体上更适合于动态关键词和空间位置关系的索引.

- 集合 SKQ

集合空间关键词查询(collective spatial keyword queries,简称 CoSKQ)[44,45]定义如下:给定一个查询 $q=\langle loc, kws\rangle$,找出一个集合 S,使得 S 覆盖了 $q.kws$ 中的关键词,而且代价最低.代价可以有多种定义:

- 代价 1: $Cost(q,S)=\alpha\cdot\sum_{o\in S} dist(q,o)+(1-\alpha)\cdot\max_{o_1,o_2\in S} dist(o_1,o_2)$.
- 代价 2: $Cost(q,S)=\alpha\cdot\max_{o\in S} dist(q,o)+(1-\alpha)\cdot\max_{o_1,o_2\in S} dist(o_1,o_2)$.
- 代价 3: $Cost(q,S)=\max_{o_1,o_2\in S\cup\{q\}} dist(o_1,o_2)$.

代价 1 和代价 2 均考虑了 S 中所有对象到 q 的空间距离以及 S 中组内距离,而代价 3 则定义为对象(含查询)之间的最大距离.无论采用何种代价模型,可以证明,CoSKQ 是一个 NP 完全问题.针对代价 1 和代价 2,文献[44]基于 IR-tree 索引提出了一种精确算法,适合于关键词数量不多的场合;还提出了一种近似算法,并证明了近似算法的代价是有界的.针对代价 2 和代价 3,文献[45]发现:给定一组对象集合,其代价最多由其中 3 个对象决

定.基于这一观察,文献提出了一个效率更高的精确算法以及一个更好的近似算法,提高了算法的近似度.

文献[46]将 CoSKQ 查询处理的问题扩展到了分布式环境中,基于 Spark 框架提出了一个分布式解决方案,使得方案具有可扩展性.

- 联合 SKQ(joint SKQ)

文献[12]研究了多 SKQ 的联合执行问题,基于 WIR-tree 提出了高效的处理算法 GROUP.对于逐个 SKQ 处理来说,该算法大幅提高了效率.

- 方向感知的 SKQ

文献[11]研究了方向感知的 SKQ,该查询除了包含位置和关键词外,还指定了方向区间,要求返回离查询者最近且覆盖了所有关键词,且在指定方向区间的 k 个对象.

- 逆空间-文本 kNN 查询

逆空间-文本 kNN 查询(reverse spatial and textual k-nearest neighbor,简称 RSTkNN)查找这样的对象:它们以查询对象为 k 最相似(空间-文本相似)对象之一,其中,空间-文本相似度是空间距离和文本相似度的线性组合.文献[47]提出了 IUR-tree(intersection-union r-tree)以及高效算法以支持这种查询.

- 空间-文本相似度搜索

在空间-文本相似度搜索(spatio-textual similarity search)[48]中,每个查询带有一个位置和关键词集合,要求返回与查询相似的对象的集合.其中,相似是指空间相似性和文本相似性均超过指定的相似度阈值.

- 交互式空间关键词搜索

文献[19]提出用交互的方式引导用户表达空间关键词查询,而不要求用户直接给出排序公式.系统与用户进行若干轮的交互,每一轮有策略地给出若干对象,交由用户选择;然后,系统学习用户偏好,最终根据学到的偏好检出结果.

5 总结与展望

本文对空间关键词查询进行了分类,然后,从索引技术和查询算法两个方面介绍了空间关键词查询处理技术.目前的研究已经取得了阶段性的成果,但是还存在值得进一步研究的问题:

(1) 动态环境下的空间关键词搜索问题

目前主要的工作均针对查询和对象的位置、文本为静态的情形,研究者已经做了一定的扩展,考虑了连续(移动)空间关键词搜索.这些研究假设查询目标处于静态,而查询者为动态.另外还有一些场景,其中,查询目标处于移动状态,查询者处于静止或者移动状态,对这些场景的研究目前较为匮乏.此外,对象的文本也可能发生变化.考虑一个微博用户,用户的位置和发表的微博都在动态更新,这种情形下,如何实现空间关键词搜索,对于索引技术有很高的要求.

(2) 考虑更多的用户偏好

在实际场景中,用户在查询一个空间-文本对象时往往有更多的偏好,例如查询餐厅时,用户可能存在对于服务、卫生等的偏好.这种偏好有时可以通过关键词表达出来,有时是通过谓词表达出来.如,假定餐厅存在服务水平和卫生程度的评分,那么这种偏好可以表达为“服务水平>4.0”或者“卫生程度>4.5”.如何处理这些偏好,使得查询结果更好地满足用户需求,是一个非常有意义的研究问题.文献[49]在这方面进行了探讨.

(3) 考虑时间约束

当同时考虑位置、文本以及时间时,可以提出很多有意义的查询.目前的研究做了初步尝试,但还远远不够.把时间引入 SKQ 后,问题变得更加复杂:在技术上涉及到空间数据库、关键词查询和时态数据库的技术,剪枝时要考虑的约束更多.如何面对更加复杂的约束和集成更加多样的技术,是很大的挑战.

(4) 查询结果的多样化

现有的文献一般根据空间距离和文本相关性返回一个对象列表,但是对于实际用户而言,它们更希望看到多种不同类型的结果,即多样化的结果.如,当用户查找最近的餐厅时,与其返回多个中餐厅,不如返回多个不同

类型的餐厅,如中餐厅、西餐厅、快餐店等,这样,用户可以有更多的选择余地.目前的文献缺乏对这一问题的考虑.

(5) 可伸缩性

在大数据背景下,很多空间-文本数据集的大小增长非常快,此时,可伸缩性就显得非常重要.目前的技术对这一问题考虑得还不够,所提出的技术可伸缩性较差.

(6) 各种技术的评测

文献[35]对主流的技术进行了实验比较,但是不够全面.评测可以从以下方面进行改进:扩充比较对象使得更加全面,实验设置考虑更多场合,设置指标更加全面.

References:

[1] Zhou AY, Yang B, Jin C, Ma Q. Location-Based services: Architecture and progress. Chinese Journal of Computers, 2011,34(7): 1155−1171 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01155]

[2] Miller R. 24% of all Google searches have local Intent. 2012. http://www.advancessg.com/43-percent-google-searches-have-local-intent/

[3] Land SE. Microsoft: 53 percent of mobile searches have local Intent. 2010. http://searchengineland.com/microsoft-53-percent-of-mobile-searcheshave-local-intent-55556

[4] Zhou YH, Xie X, Wang C, GongYC, Ma WY. Hybrid index structures for location-based Web search. In: Proc. of the CIKM. New York: ACM Press, 2005. 155−162. [doi: 10.1145/1099554.1099584]

[5] Chen YY, Suel T, Markowetz A. Efficient query processing in geographic Web search engines. In: Proc. of the ACM SIGMOD. New York: ACM Press, 2006. 277−288. [doi: 10.1145/1142473.1142505]

[6] Christoforaki M, He J, Dimopoulos C, Markowetz A, Suel T. Text vs. space: Efficient geo-search query processing. In: Proc. of the CIKM. New York: ACM Press, 2011. 423−432. [doi: 10.1145/2063576.2063641]

[7] Li ZS, Lee KCK, Zheng BH, Lee WC, Lee DL, Wang XF. IR-Tree: An efficient index for geographic document search. IEEE Trans. on Knowledge and Data Engineering, 2011,23(4):585−599. [doi: 10.1109/TKDE.2010.149]

[8] Zhang CY, Zhang Y, Zhang WJ, Lin XM. Inverted linear quadtree: Efficient top *K* spatial keyword search. In: Proc. of the ICDE. Washington: IEEE, 2013. 901−912. [doi: 10.1109/ICDE.2013.6544884]

[9] Cary A, Wolfson O, Rishe N. Efficient and scalable method for processing top-*k* spatial Boolean queries. In: Proc. of the SSDBM. LNCS 6187, Berlin, Heidelberg: Springer-Verlag, 2010. 87−95. [doi: 10.1007/978-3-642-13818-8_8]

[10] Felipe ID, Hristidis V, Rishe N. Keyword search on spatial databases. In: Proc. of the ICDE. Washington: IEEE, 2008. 656−665. [doi: 10.1109/ICDE.2008.4497474]

[11] Li GL, Feng JH, Xu J. DESKS: Direction-aware spatial keyword search. In: Proc. of the ICDE. Washington: IEEE, 2012. 474−485. [doi: 10.1109/ICDE.2012.93]

[12] Wu DM, Yiu ML, Cong G, Jensen CS. Joint top-*K* spatial keyword query processing. IEEE Trans. on Knowledge and Data Engineering, 2012,24(10):1889−1903. [doi: 10.1109/TKDE.2011.172]

[13] Cong G, Jensen CS, Wu DM. Efficient retrieval of the top-*k* most relevant spatial Web objects. Proc. of the VLDB Endowment, 2009, 2(1):337−348. [doi: 10.14778/1687627.1687666]

[14] Huang WH, Li GL, Tan KL, Feng JH. Efficient safe-region construction for moving top-*K* spatial keyword queries. In: Proc. of the CIKM. New York: ACM Press, 2012. 932−941. [doi: 10.1145/2396761.2396879]

[15] Rocha-Junior JB, Gkorgkas O, Jonassen S, Nørvåg K. Efficient processing of top-*k* spatial keyword queries. In: Proc. of the 12th Int'l Conf. on Advances in Spatial and Temporal Databases. Berlin, Heidelberg: Springer-Verlag, 2011. 205−222. [doi: 10.1007/978-3-642-22922-0_13]

[16] Rocha-Junior JB, Nørvåg K. Top-*k* spatial keyword queries on road networks. In: Proc. of the EDBT. New York: ACM Press, 2012. 168−179. [doi: 10.1145/2247596.2247617]

[17] Wu DM, Yiu ML, Jensen CS, Cong G. Efficient continuously moving top-*k* spatial keyword query processing. In: Proc. of the ICDE. Washington: IEEE, 2011. 541−552. [doi: 10.1109/ICDE.2011.5767861]

[18] Zhang DX, Tan KL, Tung AKH. Scalable top-*k* spatial keyword search. In: Proc. of the EDBT. New York: ACM Press, 2013. 359−370. [doi: 10.1145/ 2452376.2452419]

[19] Zheng K, Su H, Zheng BL, Shang S, Xu JJ, Liu JJ, Zhou XF. Interactive top-*k* spatial keyword queries. In: Proc. of the ICDE. Washington: IEEE, 2015. 423−434. [doi: 10.1109/ICDE.2015.7113303]

[20] Zhang DX, Chan CY, Tan KL. Processing spatial keyword query as a top-*k* aggregation query. In: Proc. of the SIGIR. New York: ACM Press, 2014. 355−364. [doi: 10.1145/2600428.2609562]

[21] Wu DM, Cong G, Jensen CS. A framework for efficient spatial Web object retrieval. The VLDB Journal, 2012,21(6):1−26. [doi: 10.1007/s00778-012-0271-0]

[22] Vaid S, Jones CB, Joho H, Sanderson M. Spatio-Textual indexing for geographical search on the Web. In: Proc. of the Int'l Conf. on Advances in Spatial and Temporal Databases. Berlin, Heidelberg: Springer-Verlag, 2005. 218−235. [doi: 10.1007/11535331_13]

[23] Hariharan R, Hore B, Li C, Mehrotra S. Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In: Proc. of the SSDBM. Washington: IEEE, 2007. 16−25. [doi: 10.1109/ssdbm.2007.22]

[24] Hu J, Fan J, Li GL, Chen SS. Top-*k* fuzzy spatial keyword search. Chinese Journal of Computers, 2012,35(11):2237−2246 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.02237]

[25] Yao B, Li, FF, Hadjieleftheriou M, Hou K. Approximate string search in spatial databases. In: Proc. of the ICDE. Washington: IEEE, 2010. 545−556. [doi: 10.1109/ICDE.2010.5447836]

[26] Wang JB, Gao H, Li JZ, Yang DH. An index supporting spatial approximate keyword search on disks. Journal of Computer Research and Development, 2012,49(10):2142−2152 (in Chinese with English abstract).

[27] Alsubaiee S, Behm A, Li C. Supporting location-based approximate-keyword queries. In: Proc. of the SIGSPATIAL GIS. New York: ACM Press, 2010. 61−70. [doi: 10.1145/1869790.1869802]

[28] Guttman A. R-trees: A dynamic index structure for spatial searching. SIGMOD Record, 1984,14(2):47−57. [doi: 10.1145/971697.602266]

[29] Beckmann N, Kriegel HP, Schneider R, Seeger B. The R*-tree: An efficient and robust access method for points and rectangles. In: Proc. of the ACM SIGMOD. New York: ACM Press, 1990. 322−331. [doi: 10.1145/93597.98741]

[30] Zhang DX, Ooi BC, Tung AKH. Locating mapped resources in Web 2.0. In: Proc. of the ICDE. Washington: IEEE, 2010. 521−532. [doi: 10.1109/ICDE.2010.5447897]

[31] Zhang DX, Chee YM, Mondal A, Tung AKH, Kitsuregawa M. Keyword search in spatial databases: Towards searching by document. In: Proc. of the ICDE. Washington: IEEE, 2009. 688−699. [doi: 10.1109/icde.2009.77]

[32] Fagin R, Lotem A, Naor M. Optimal aggregation algorithms for middleware. In: Proc. of the PODS. New York: ACM Press, 2001. 102−113. [doi: 10.1145/ 375551.375567]

[33] Chen C, Chen CN, Sun WW. Spatial keyword queries in wireless broadcast environment. Journal of Computer Research and Development, 2013,50(S1):145−153 (in Chinese with English abstract).

[34] Hjaltason GR, Samet H. Distance browsing in spatial databases. ACM Trans. on Database System, 1999,24(2):265−318. [doi: 10.1145/320248.320255]

[35] Chen L, Cong G, Jensen CS, Wu DM. Spatial keyword query processing: An experimental evaluation. 2013. Proc. of the VLDB Endowment, 2013,6(3):217−228. [doi: 10.14778/2535569.2448955]

[36] Hu HB, Xu JL, Lee DL. A generic framework for monitoring continuous spatial queries over moving objects. In: Proc. of the ACM SIGMOD. New York: ACM Press, 2005. 479−490. [doi: 10.1145/1066157.1066212]

[37] Prabhakar S, Xia Y, Kalashnikov DV, Aref WG, Hambrusch SE. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. IEEE Trans. on Computers, 2002,51(10):1124−1140. [doi: 10.1109/TC.2002.1039840]

[38] Guo L, Shao J, Aung HH, Tan KL. Efficient continuous top-*k* spatial keyword queries on road networks. Geoinformatica, 2014, 19(1):29−60. [doi: 10.1007/s10707-014-0204-8]

[39] Li YH, Li GH, Zhang C. Processing continuous top-*k* spatial keyword queries over road networks. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2013,41(12):54−58 (in Chinese with English abstract).

[40] Skovsgaard A, Sidlauskas D, Jensen CS. Scalable top-*k* spatio-temporal term querying. In: Proc. of the ICDE. Washington: IEEE, 2014. 148−159. [doi: 10.1109/ICDE.2014.6816647]

[41] Nepomnyachiy S, Gelley B, Jiang W, Minkus T. What, where, and when: Keyword search with spatio-temporal ranges. In: Proc. of the 8th Workshop on Geographic Information Retrieval. New York: ACM Press, 2014. 1−8. [doi: 10.1145/2675354.2675358]

[42] Chen LS, Cong G, Cao X, Tan KL. Temporal spatial-keyword top-*k* publish/subscribe. In: Proc. of the ICDE. Washington: IEEE, 2015. 255−266. [doi: 10.1109/ICDE.2015.7113289]

[43] Dai J, Xu JJ, Liu KE, Wu B, Ding ZM. DKR-Tree: A dynamic-keyword-R tree. Journal of Computer Research and Development, 2013,50(S1):163−170 (in Chinese with English abstract).

[44] Cao X, Cong G, Jensen CS, Ooi BC. Collective spatial keyword querying. In: Proc. of the ACM SIGMOD. New York: ACM Press, 2011. 373−384. [doi: 10.1145/1989323.1989363]

[45] Long C, Wong RC, Wang K, Fu AWC. Collective spatial keyword queries: A distance owner-driven approach. In: Proc. of the SIGMOD. New York: ACM Press, 2013. 689−700. [doi: 10.1145/2463676.2465275]

[46] He PJ, Xu H, Zhao X, Shen ZT. Scalable collective spatial keyword query. In: Proc. of the 31st IEEE Int'l Conf. on Data Engineering Workshops (ICDEW). Washington: IEEE, 2015. 182−189. [doi: 10.1109/ICDEW.2015.7129574]

[47] Lu JH, Lu Y, Cong G. Reverse spatial and textual *k* nearest neighbor search. In: Proc. of the ACM SIGMOD. New York: ACM Press, 2011. 349−360. [doi: 10.1145/1989323.1989361]

[48] Fan J, Li GL, Zhou LZ, Chen SS, Hu J. Seal: Spatio-textual similarity search. Proc. of the VLDB Endowment, 2012,5(9):824−835. [doi: 10.14778/2311906.2311910]

[49] Liu XP, Chen L, Wan CX. LINQ: A framework for location-aware indexing and query processing. IEEE Trans. on Knowledge and Data Engineering, 2015,27(5):1288−1300. [doi: 10.1109/TKDE.2014.2365792]

附中文参考文献:

[1] 周傲英,杨彬,金澈清,马强.基于位置的服务:架构与进展.计算机学报,2011,34(7):1155−1171. [doi: 10.3724/SP.J.1016.2011.01155]

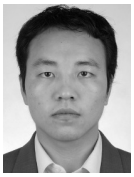
[24] 胡骏,范举,李国良,陈珊珊.空间数据上 Top-*k* 关键词模糊查询算法.计算机学报,2012,35(11):2237−2246. [doi: 10.3724/SP.J.1016.2012.02237]

[26] 王金宝,高宏,李建中,杨东华.RB 树:一种支持空间近似关键字查询的外存索引.计算机研究与发展,2012,49(10):2142−2152.

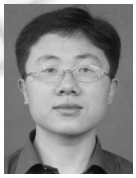
[33] 陈翀,陈楚南,孙未未.无线数据广播环境下的空间关键字查询.计算机研究与发展,2013,50(S1):145−153.

[39] 李艳红,李国徽,张聪.路网中空间关键字连续 *k* 近邻查询算法研究.华中科技大学学报(自然科学版),2013,41(12):54−58.

[43] 戴健,许佳捷,刘奎恩,武斌,丁治明.DKR-Tree:一种支持动态关键字的空间对象索引树.计算机研究与发展,2013,50(S1):163−170.

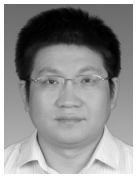


刘喜平(1981－),男,湖北红安人,博士,副教授,CCF 专业会员,主要研究领域为信息检索,数据库,数据挖掘.



刘德喜(1975－),男,博士,教授,CCF 高级会员,主要研究领域为 Web 数据管理,信息检索,社会网络.

万常选(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为 Web 信息管理,情感计算,信息检索,数据挖掘.



廖国琼(1969－),男,博士,教授,CCF 高级会员,主要研究领域为数据库,数据挖掘.