计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2014.20130827
51(10):2206-2215,2014

基于用户反馈的 top-k 查询修改算法

张建锋[1] 韩伟红[1] 樊 华[1] 邹 鹏[2] 贾 焰[1]
[1](国防科学技术大学计算机学院 长沙 410073)
[2](装备指挥学院 北京 101416)
(jfzhang@nudt.edu.cn)

An Algorithm for Top-k Query Refinement Based on User's Feedback

Zhang Jianfeng[1], Han Weihong[1], Fan Hua[1], Zou Peng[2], and Jia Yan[1]
[1](*College of Computer, National University of Defense Technology, Changsha*, 410073)
[2](*Academy of Equipment Command and Technology, Beijing*, 101416)

Abstract Top-k queries are mainly used to return the most important objects in the potentially huge answer space. After huge effort working on top-k query performance, an explain capability has attract more attentions in recent years. In top-k query, since users can not precisely specify their own preferences, he/she may feel frustrated with the top-k results and propose a question such that "why my expecting tuple m is not appeared in top-k results as long as tuple p has been appeared in top-k results". To solve this problem, a novel method is proposed to answer user's why-not question by modifying original top-k query. Firstly, an assessment model is defined to evaluate the difference between the original top-k query and the modified new query. Then based on the assessment model, a sample method is used to sample the candidate weight vectors from an accuracy subspace of weight space. After that, for each candidate weight vector, a progressive top-k algorithm is used to get the optimal new query, and the terminal conditions of the progressive top-k algorithm are improved by the assessment model. Furthermore, the new query with the best evaluated score is returned as the solution. Finally, an extensive performance study using synthetic data set is reported to verify the efficiency of the proposed algorithm.

Key words top-k query; why-not question; user's feedback; preference refinement; query refinement

摘 要 top-k 查询主要用来从海量的数据中返回用户最为偏好的 k 个对象.目前已经有大量的研究工作致力于 top-k 查询中的性能研究,近年来针对 top-k 查询结果进行解释的研究逐渐得到了广泛的关注.在 top-k 查询中,由于用户不能精确地指定自己的偏好,因此针对 top-k 查询的结果用户可能产生这样的质疑:"既然连对象 p 都出现在 top-k 结果中,为什么我期望的对象 m 块没有出现在 top-k 结果?"针对用户这样的疑问,提出了一种基于用户反馈的 top-k 查询修改算法,该算法首先定义了用来衡量初始化 top-k 查询变化的评估模型函数,基于该评估模型函数,使用抽样方法得到候选权重集合,针对每一个候选权重通过渐进式 top-k 算法来得到新的最优化查询.最后在模拟数据上验证了提出算法的效率.

关键词 top-k 查询;用户疑问;用户反馈;偏好修正;查询修改

中图法分类号 TP311.13; TP393

收稿日期:2013-06-18;**修回日期**:2013-07-05

基金项目:国家自然科学基金项目(60933005);国家"八六三"高技术研究发展计划基金项目(2011AA010702,2012AA01A401,2012AA01A402)

top-k 查询主要用来从海量的数据中返回用户最为偏好的 k 个对象.目前已经有大量的研究工作致力于 top-k 查询中性能研究,近年来针对 top-k 查询结果进行解释的研究逐渐得到了广泛的关注.比如在大规模网络安全态势评估中,针对不同的使用需求,为了从海量的网络安全告警中返回给用户最具有威胁的 k 个告警,系统基于用户在告警属性上面的偏好使用 top-k 查询来返回用户最具有威胁性的 k 个告警.但是针对返回的 k 个网络安全告警信息,用户可能提出"为什么"的疑问,即用户在返回的告警分析的基础上可能会抛出这样的疑问"我认为告警 m 应该比告警 p 严重一些,为什么告警 p 出现在最具有威胁的 k 个告警结果中,而告警 m 没有出现?",上述的疑问表明在 top-k 查询中所设置的查询参数(参数 k 和用户偏好)可能与用户的期望值有一定差距.针对该问题,基于用户的反馈,需要研究如何来回答用户提出的"为什么"的问题.如表 1 所示,用户需要从 5 个网络安全告警中得到最具有威胁的 2 个网络安全告警,因此使用加权求和函数对这 5 个告警进行评分.设置初始化偏好为[0.333,0.333,0.334],在该设置下最具有威胁的 2 个告警为告警 5 和告警 4.不幸的是用户对系统返回的这个结果不是特别满意,在用户看来告警 1 应该比告警 4 更具有威胁性,为何告警 4 出现在最具有威胁的 2 个告警中,而告警 1 没有出现呢?

Table 1 Network Security Alerts

表 1 网络安全告警

Alert ID	Asset	Threat	Frequency
1	0.7	0.3	0.1
2	0.2	0.4	0.7
3	0.5	0.2	0.2
4	0.4	0.8	0.5
5	0.8	0.6	0.8

为了解决上述的问题,本文提出了基于用户反馈的查询修改方法.一般而言 top-k 查询主要取决于 2 个参数:参数 k 和参数 $\boldsymbol{w}$,其中参数 k 控制返回结果集大小,参数 $\boldsymbol{w}$ 表示用户在数据各个属性上面的偏好.因此为了解释用户针对排序结果提出的疑问,需要基于用户的反馈对查询的参数进行调整.为了解决这个问题,首先本文定义了一个评估模型来衡量新查询与初始化查询结果之间的差异性.基于该评估模型本文试图返回给用户一个新的查询,该新查询既能有效地回答用户的疑问,又能使得初始化查询结果的变化最小.

本文的主要贡献如下:

1) 介绍了本文所研究问题的背景和来源,在此基础上形式化的定义了本文所要研究的问题;

2) 提出了一种衡量 2 次查询之间差异性的评估模型.在该模型下基于用户的反馈提出了一种得到最优化新查询的近似算法,并从多个方面着手提高该算法的执行效率;

3) 在实现该算法的基础上,使用模拟数据集对算法的执行效率进行了测试.

1 问题定义与描述

本节首先介绍了一些基础性的工作,然后形式化地定义了 top-k 查询中的"为什么"问题.

1.1 相关概念的形式化定义

数据库 D 包含 n 个对象,每一个对象通过 d 个属性字段进行描述.$p.A_i$ 用来描述对象 p 的属性 A_i 字段的值.为了方便后面的讨论,假设所有属性字段均为数字型属性且所有属性值均被归一化到 0~1 的区间.不失一般性,假设用户更偏好喜欢属性值大的对象.

top-k 查询主要用来从海量的数据中返回用户最为偏好的 k 个对象.通过一个用户定义的偏好性函数 f 来对数据库中的每一个对象进行打分,分值最高的 k 个对象作为用户 top-k 查询的结果返回给用户.最常用的偏好性函数 f 为线性聚类函数,比如加权求和函数.在线性加权求和函数下,用户为每一个属性 d_i 指定一个权重 $w[i]$,每一个对象的分值是由该对象各个属性值组成的向量和各属性上权重 $w[i]$组成的向量 $\boldsymbol{w}$ 的点积计算而得,即 $score(\boldsymbol{p},\boldsymbol{w})=\boldsymbol{p}\cdot\boldsymbol{w}$.因此 top-$k$ 查询的结果为 k 个评分值最高的对象的有序列表.

top-k 查询:给定一个正整数 k 和一个权重向量 $\boldsymbol{w}$,top-k 查询 $Q(k,\boldsymbol{w})$的结果为一个有序的对象序列,该对象序列满足 $Q(k,\boldsymbol{w})\in D$,$|Q(k,\boldsymbol{w})|=k$ 且 $\forall p_1,p_2:p_1\in Q(k,\boldsymbol{w}),p_2\in D-Q(k,\boldsymbol{w})$ 均有 $score(\boldsymbol{p}_1,\boldsymbol{w})>score(\boldsymbol{p}_2,\boldsymbol{w})$.

在欧氏几何空间里,一个线性的 top-k 查询可以简单用向量 $\boldsymbol{w}$ 来表示.文献[1]中表明在向量 $\boldsymbol{w}$ 角度保持不变的前提下,$\boldsymbol{w}$ 的长度对 top-k 查询的结果不产生任何影响.因此为了方便讨论,假设向量 $\boldsymbol{w}$ 为单位长度向量,即 $\sum_{i=1}^{d}w[i]=1$.

1.2 问题定义

第 k 个对象:给定一个正整数 k 和一个权重向量 $\boldsymbol{w}$,第 k 个对象 $Tuple(k,\boldsymbol{w})$ 为数据库里的一个对象 p,且该对象满足 $p\in Q(k,\boldsymbol{w})-Q(k-1,\boldsymbol{w})$.

在开始阶段,用户给定一个 top-k 的查询 $Q_0(k_0,\boldsymbol{w}_0)$. 基于该查询的返回结果,用户可能提出这样一个疑问"对象 p 都出现在 top-k 的结果中了,为什么对象 m 没有出现?"在回答这个问题之前,首先我们假定对象 m 在数据库中是真实存在的,其次 p 和 m 是不可比较的,即不存在对象 p 支配 m 或对象 m 支配 p. 基于这些假设,为了回答用户的"为什么"问题,需要对初始化的查询 $Q_0(k_0,\boldsymbol{w}_0)$ 进行修改,修改后得到新查询 $Q'(k',\boldsymbol{w}')$ 是"为什么"问题的解的充要条件为:

1) 用户期望的对象 m 出现在新查询 $Q'(k',\boldsymbol{w}')$ 的结果中;

2) $score(\boldsymbol{m},\boldsymbol{w}')>score(\boldsymbol{p},\boldsymbol{w}')$.

通过该新查询来解释用户的疑问,最终将用户对 top-k 结果的期望通过参数修改的方法作用到系统初始化设置的参数上面,使得系统在后续的 top-k 查询过程中能够体现出用户的具体使用需求. 显然,满足上述 2 个条件的新的查询有很多,因此需要定义一个评估模型来衡量得到的新的查询效率,该评估模型定义如下:

$$Eval(Q(k',\boldsymbol{w}'))=\lambda_w\Delta W+\lambda_c\Delta C, \tag{1}$$

其中,$\Delta W=\|\boldsymbol{w}'-\boldsymbol{w}_0\|_2$ 用来表示 2 个查询向量之间的差异性,ΔC 用来衡量 2 次查询结果之间的差异性,λ_w,λ_c 分别用来衡量用户对修改查询向量和查询结果的容忍度,且 $\lambda_w+\lambda_c=1$. 查询结果之间的差异性 ΔC 计算如下:

$$\Delta C=|Q(k_0,\boldsymbol{w}_0)\bigcup Q(k',\boldsymbol{w}')|-|Q(k_0,\boldsymbol{w}_0)\bigcap Q(k',\boldsymbol{w}')|.$$

因为 ΔC 远远大于 ΔW,式(1)需要进一步的归一化,具体的方法见 3.3.1 节. 通过评估模型式(1)可以看出,具有较小评价值的新查询具有较高的效率,因为其对原始查询 $Q_0(k_0,\boldsymbol{w}_0)$ 的改动最小.

图 1 所示为三维空间下面的一个示例. 假设用户初始化设置的偏好为 $\boldsymbol{w}_0=[0.333,0.333,0.334]$,在这个偏好下,$Q(2,\boldsymbol{w}_0)$ 的返回结果是告警 5 和告警 4. 基于这个返回结果,用户可能会提出这样一个问题"告警 4 都出现在 top-2 的结果中了,为何没有告警 1?"这个疑问表明在用户的潜意识里面可能认为告警 1 比告警 4 更加严重一些. 如图 1 所示,告警 1 和告警 4 形成的直线 $(0.7-0.4)w[1]+(0.3-0.8)w[2]+(0.1-0.5)(1-w[1]-w[2])=0$ 将权重空间分割成两部分,用户潜意识里的偏好(即告警 1 比告警 4 严重)位于该直线的右下方. 假设从该区域抽样得到 2 个权重向量 $\boldsymbol{w}_1$ 和 $\boldsymbol{w}_2$,在 $\boldsymbol{w}_1$ 下 5 个告警严重程度排序为告警 5,1,4,3,2,而在 $\boldsymbol{w}_2$ 下 5 个告警严重程度排序为告警 5,1,3,4,2. 对 $\boldsymbol{w}_1$ 而言,新查询 $Q(2,\boldsymbol{w}_1)$ 和 $Q(3,\boldsymbol{w}_1)$ 都可以回答用户提出的"为什么"的问题,但是 $Q(3,\boldsymbol{w}_1)$ 相对于 $Q(2,\boldsymbol{w}_1)$ 而言仅仅在原始查询 $Q(2,\boldsymbol{w}_0)$ 的基础上增加了告警 1,因此我们认为解 $Q(3,\boldsymbol{w}_1)$ 优于 $Q(2,\boldsymbol{w}_1)$. 对 $\boldsymbol{w}_2$ 而言,同理可得 $Q(2,\boldsymbol{w}_2)$ 为该权重空间下面的一个最优解. 最后通过评价模型式(1),可以得到 $Q(3,\boldsymbol{w}_1)$ 优于 $Q(2,\boldsymbol{w}_2)$. 因此将 $Q(3,\boldsymbol{w}_1)$ 作为回答用户"为什么"问题的最优解返回给用户.

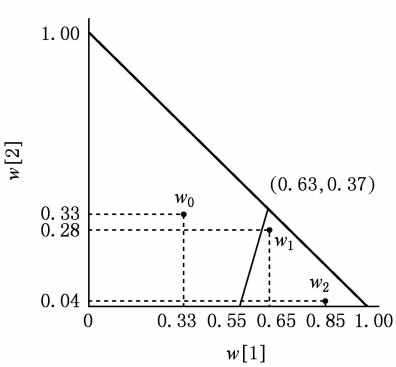


Fig. 1 Example of "why-not" question under 3D.

图 1 三维下面用户"为什么"示例

1.3 问题分析

通过 1.2 节对问题的定义和讨论,为了回答用户"为什么"的问题需要在原有查询的基础上修改得到一个新的查询,该新查询必须满足用户期望对象 m 必须排在对比对象 p 的前面,即 $score(\boldsymbol{m},\boldsymbol{w}')>score(\boldsymbol{p},\boldsymbol{w}')$. 在数据空间里,如果对象 m 支配对象 p,那么意味着 $score(\boldsymbol{m},\boldsymbol{w}')>score(\boldsymbol{p},\boldsymbol{w}')$ 永远成立;如果对象 p 支配对象 m,那么意味着 $score(\boldsymbol{p},\boldsymbol{w}')>score(\boldsymbol{m},\boldsymbol{w}')$ 永远成立. 因此假设本文中用户提出的"为什么"问题中对象 p 和对象 m 是不可比较的,即对象 p 和对象 m 之间不存在支配关系.

基于不可比较的数据对象 p 和 m,权重空间的一个超平面 $H:(\boldsymbol{m}-\boldsymbol{p})\cdot\boldsymbol{w}=0$ 将权重空间分割成 $W_>$ 和 $W_<$ 两部分. 位于 $W_>$ 区域的所有权重向量 $\boldsymbol{w}$ 使得 $score(\boldsymbol{m},\boldsymbol{w})>score(\boldsymbol{p},\boldsymbol{w})$. 位于 $W_<$ 区域的所有权重向量 $\boldsymbol{w}$ 使得 $score(\boldsymbol{m},\boldsymbol{w})<score(\boldsymbol{p},\boldsymbol{w})$. 因此为了得到满足条件的新查询,仅仅需要考虑权重空间的子空间 $W_>$ 即可. 针对该子空间的每一个权重

向量 $\boldsymbol{w}$ 均可以得到一个满足所有条件的最优新查询.在这个基础上,针对所有权重向量的最优新查询,通过评估模型式(1),可以得到各个权重向量下最优解的效率.因此可以将具有最高效率的新查询作为最终的解用来回答用户的"为什么"问题.但是在权重空间 $W_>$ 里面存在着无限多个可能的权重向量 $\boldsymbol{w}$,因此很难通过穷举所有的权重向量来得到问题的最优解.所以本文提出了一种新的优化算法来回答用户"为什么"的问题.

2 相关研究

针对用户对查询结果的疑问,Chapman 等人[2]首先提出了解释查询过程中的"为什么"问题这个概念.在这之前,文献[3]针对数据库查询过程中结果集为空的现象进行了解释,但是这些研究都没有具体地提出如何解决查询过程中的"为什么"问题.目前在回答用户针对查询结果疑问这个问题上代表性的工作如下:

Huang 等人[4]针对 SPJ(select-project-join)查询结果中用户期望对象缺失现象进行解释,告诉用户为什么其期望的对象没有出现在查询的结果之中;随后 Herschel 等人[5]针对 SPJUA(select-project-join-union-aggregation)查询中用户期望对象缺失现象进行解释;Tran 等人[6]针对 SPJA(select-project-join-aggregation)查询上的同一问题进行了研究.这些研究都仅限于对用户心目中期望的对象没有出现在查询结果中的现象进行解释,告诉用户到底是因为什么原因造成他们期望的对象没有出现,比如该对象不存在,或者该对象的某个属性与查询中条件不符合等.

除了针对传统的查询外,Vlachos 等人[7]提出了一种新的回答用户为什么问题方法.该问题针对基于评分函数的 top-k 查询,主要研究解决 2 个方面的内容:1)在给定数据点 q、正整数 k 和数据对集 S 的情况下,求解使得对象点 q 出现在 top-k 查询结果中的权重的集合 $mRTOP_k(q)$(monochromatic reverse top-k).该问题的解是一组权重的集合$\{\boldsymbol{w}_i\}$,在这个集合里的任意权重 $\boldsymbol{w}_i$ 下均可以使得对象 q 出现在以 k 和 $\boldsymbol{w}_i$ 为参数的 top-k 查询的结果中;2)在给定数据点 q、正整数 k、数据对象集 S 和权重集合 W 的情况下,求解集合 W 中使得对象点 q 出现在 top-k 查询结果中的权重的子集 $bRTOP_k(q)$(bichromatic reverse top-k). He 等人[8]针对用户对 top-k 查询结果疑问提出了一种新的解决的方法.在该文献中用户的输入为一个初始化 top-k 查询以及一组用户期望出现的对象集,输出为重新定义的 top-k 查询,使得用户期望的对象全部出现在该 top-k 的结果中.为了实现这个目的,主要通过修改初始化查询对应的参数 k 以及用户偏好权重 $\boldsymbol{w}$ 来实现,最终返回给用户满足要求的 top-k 查询,并且使得 k 和 $\boldsymbol{w}$ 的变化最小.但是该方法存在着明显的不足只是考虑了使得用户期望的对象出现在 top-k 结果中,而忽视对初始化 top-k 查询结果的考虑,完全脱离了原始查询去定义新的满足要求的查询.

3 基于用户反馈的 top-k 告警修正算法

本节主要介绍如何得到回答用户"为什么"问题最优解的求解方法.首先给出了基本的解决方案来说明解决该问题的核心思想,随后分析和给出解决该问题的一些优化策略.

3.1 基本解决方案

本节详细介绍了基本的解决方案.基本的解决方案主要分为以下 3 个关键步骤:

步骤 1. 初始化权重空间 $\Gamma=\{(\boldsymbol{m}-\boldsymbol{p})\cdot\boldsymbol{w}>0, w[i]\in[0,1], \sum_{i=1}^{d} w[i]=1\}$,从该空间里抽样得到 s 个候选权重向量 $S=\{\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_s\}$;

步骤 2. 基于候选集合 S,针对每一个权重向量 $\boldsymbol{w}_i\in S$,利用渐进式 top-k 计算方法[9-10]得到每一个新的查询;

步骤 3. 针对新得到所有查询,利用评估模型式(1)进行评估,选择效率最高的新查询作为回答用户"为什么"问题的查询返回给用户.

接下来将对上述基本算法的效率进行优化.

3.2 渐进式 top-k 算法的终止条件

本节主要讨论在给定的权重向量 $\boldsymbol{w}_i\in S$ 下,如何得到渐进式 top-k 的终止条件.渐进式 top-k 算法是一种增量式计算 top-k 方法,算法每执行一步,返回一个最优的对象给用户.渐进式 top-k 算法终止的条件是所有的 k 个对象全部产生.那么针对 $\boldsymbol{w}_i$,如何得到参数 k 呢?

按照 1.2 节的讨论,针对给定的 $\boldsymbol{w}_i$,需要得到最优的新查询使得评估模型(式 1)值最小.针对给定的 $\boldsymbol{w}_i$,不同的新查询具有相同的 ΔW.所以要使得式(1)值最小,必须使 ΔC 的值最小.根据求解 ΔC 的公式得知,ΔC 完全依赖于初始化查询 $Q_0(k_0,\boldsymbol{w}_0)$ 和新查询 $Q(k',\boldsymbol{w}_i)$.因为 $Q_0(k_0,\boldsymbol{w}_0)$ 和 $\boldsymbol{w}_i$ 已经完全确定,所以 ΔC 的大小完全取决于 k' 的大小.为了方

便讨论,这里 ΔC_k 用来表示通过新查询 $Q(k,\boldsymbol{w}_i)$ 和初始化查询 $Q_0(k_0,\boldsymbol{w}_0)$ 计算而得到的 ΔC.

算法 1. 最优 k 的求解算法.

输入:数据集 D、权重向量 $\boldsymbol{w}$、用户的初始化查询 $Q_0(k_0,\boldsymbol{w}_0)$、对象 m、对象 p;

输出:新查询 $Q(k,\boldsymbol{w})$.

① $Q(k_m,\boldsymbol{w})$←执行渐进式 top-k 算法直到 m 出现;

② $a=|q|q\in Q(k_m,\boldsymbol{w})\cap Q_0(k_0,\boldsymbol{w}_0)|$;

③ $k_{\text{ub}}\leftarrow k_m+2(k_0-a)$;

④ $\Delta C=\Delta C_{\min}=k_0+k_m-2a$;

⑤ $k_{\min}=k_m$;

⑥ $k=k_m+1$;

⑦ while $k\leqslant k_{\text{ub}}$ do

⑧ if $Tuple(k,\boldsymbol{w})\in Q_0(k_0,\boldsymbol{w}_0)$ then

⑨ $\Delta C=\Delta C-1$;

⑩ else

⑪ $\Delta C=\Delta C+1$;

⑫ end if

⑬ if $\Delta C<\Delta C_{\min}$ then

⑭ $\Delta C_{\min}=\Delta C$;

⑮ $k_{\min}=k$;

⑯ end if

⑰ $k=k+1$;

⑱ return $Q(k_{\min},\boldsymbol{w})$;

算法 1 给出了计算给定 $\boldsymbol{w}_i$ 下面最优的参数 k 的解决方法,下面将给出本算法的正确性分析.为了证明算法 1 的正确性,需要证明在给定 $\boldsymbol{w}$ 下,使得式(1)取得最小值的 k 存在上界.

定理 1. 给定权重空间 $\boldsymbol{w}$,渐进式 top-k 算法可以在 $k_m+2(k_0-a)$ 步骤内找到使得式(1)取得最小值的 k,其中 k_m 为满足 $Tuple(k_m,\boldsymbol{w})=m$,$a$ 为 $Q(k_m,\boldsymbol{w})$ 和 $Q_0(k_0,\boldsymbol{w}_0)$ 共同对象的个数.

证明. 该定理通过下面 2 种情况来证明:

1) $Q(k_m,\boldsymbol{w})$ 可能不是最优的解,因此需要继续执行渐进式 top-k 算法;

2) 使得式(1)取得最小值的 k 存在上界 $k_m+2(k_0-a)$.

情况 1)的证明可以通过构造一个例子来说明.如果 $Tuple(k_m+1,\boldsymbol{w})\in Q_0(k_0,\boldsymbol{w}_0)$,那么 ΔC 的值随着 k 的增加而减小,即 $Q(k_m+1,\boldsymbol{w})$ 在评估模型式(1)下面优于 $Q(k_m,\boldsymbol{w})$.因此这种情况表明即使用户期望的对象 m 已经出现在 top-k 的结果中,为了得到最优解,渐进式 top-k 算法仍然不能终止.

情况 2)可以通过反证法来证明,只需要证明对于 $\forall k''>k_m+2(k_0-a)$,均有 $\Delta C_{k''}>\Delta C_{k_m}$ 即可.这样当 k'' 大于设置的上界时,任何新查询的效率都低于已有的新查询 $Q(k_m,\boldsymbol{w})$.证明过程如下:

$$\Delta C_{k''}=|Q(k_0,\boldsymbol{w}_0)\cup Q(k'',\boldsymbol{w})|-|Q(k_0,\boldsymbol{w}_0)\cap Q(k'',\boldsymbol{w})|=(k_0+k''-a'')-a''=k_0+k''-2a'',$$

$$\because\quad k''>k_m+2(k_0-a),$$

$$\therefore\quad \Delta C_{k''}>k_0+k_m+2(k_0-a)-2a'',$$

$$\because\quad a''\leqslant k_0,$$

$$\therefore\quad \Delta C_{k''}\geqslant k_0+k_m-2a=k_0+(k_m-a)-a=|Q(k_0,\boldsymbol{w}_0)\cup Q(k_m,\boldsymbol{w})|-|Q(k_0,\boldsymbol{w}_0)\cap Q(k_m,\boldsymbol{w})|=\Delta C_{k_m},$$

其中,$a''=|q|q\in Q(k'',\boldsymbol{w})\cap Q_0(k_0,\boldsymbol{w}_0)|$.通过情况 2)可以得出 $Eval(Q(k_m,\boldsymbol{w}))<Eval(Q(k'',\boldsymbol{w}))$ 永远成立.因此当 $k>k_m+2(k_0-a)$ 时,渐进式 top-k 算法可以安全的被终止.

综上所述,算法 1 的正确性得到了有效地保证.

证毕.

3.3 权重空间的删减策略

本节详细讨论如何得到候选的权重向量集合 S.在 3.1 节中,为了回答用户的"为什么"的问题,从权重子空间抽样得到候选权重集合 S.该权重子空间通过下面一组不等式来描述 $\Gamma=\{(\boldsymbol{m}-\boldsymbol{p})\cdot\boldsymbol{w}>0,w[i]\in[0,1],\sum_{i=1}^{d}w[i]=1\}$.结合评估模型式(1),抽样权重子空间可以进一步的缩小,从而进一步提高抽样的效率.具体步骤如下:

1) 首先将 $\boldsymbol{w}_0$ 投影到权重空间 Γ 上,求得使得式(1)中 ΔW 最小的投影点 $\boldsymbol{w}_1$[11];

2) 然后基于投影点 $\boldsymbol{w}_1$,利用算法 1 求得回答用户"为什么"问题的第 1 个解 $Q(k_1,\boldsymbol{w}_1)$,并得到其新查询的评估值 $E_1=\lambda_w\Delta W_1+\lambda_c\Delta C_1$;

3) 基于新查询 $Q(k_1,\boldsymbol{w}_1)$ 的评估值 E_1,进一步得到 ΔW 的最大值 $\Delta W_{\max}=E_1/\lambda_w$,从而对原始的权重空间进行进一步的删减,得到更加精确的抽样权重空间:$S_{\text{region}}=\{\|\boldsymbol{w}-\boldsymbol{w}_0\|_2<\Delta W_{\max}\}\cap\Gamma$.

定理 2. 对于任意 $\boldsymbol{w}_i\in W-S_{\text{region}}$,$Eval(Q(k_1,\boldsymbol{w}_1))<Eval(Q(k_i,\boldsymbol{w}_i))$ 都成立.

证明. 该定理的具体证明过程如下:

$$Eval(Q(k_i,\boldsymbol{w}_i))=\lambda_w\Delta W_i+\lambda_c\Delta C_i>\lambda_w\Delta W_{\max}+\lambda_c\Delta C_i=\lambda_w\Delta W_1+\lambda_c\Delta C_1+\lambda_c\Delta C_i=Eval(Q(k_1,\boldsymbol{w}_1))+\lambda_c\Delta C_i.$$

因为 $\lambda_c\Delta C_i>0$,所以 $Eval(Q(k_1,\boldsymbol{w}_1))<Eval(Q(k_i,\boldsymbol{w}_i))$,即从 S_{region} 区域外得到的任意权重向量,在评估模型式(1)下,其得到新查询不可能优于 $\boldsymbol{w}_1$ 得到的新查询 $Q(k_1,\boldsymbol{w}_1)$.

证毕.

基于定理 2,首先对评估模型式(1)进行进一步的改进;其次对抽样得到候选权重集合 S 的过程进行改进;最后对渐进式 top-k 算法的执行效率进行改进.

3.3.1 归一化评估模型

通过前面对评估模型的定义式(1)可以得知 ΔC 的值远远大于 ΔW 的值,因此为了使得评估模型更加合理,需要对 ΔC 和 ΔW 进行归一化处理.

针对 ΔW 而言,利用 3.3 节中求得的 $\Delta W_{\max}$ 来对 ΔW 进行归一化处理.因为候选权重集合中的权重均需要从 S_{region} 抽样得到,因此对于候选集合中的任意权重 $\boldsymbol{w}_i \in S_{\text{region}}$,其 $\Delta W = \|\boldsymbol{w}_i - \boldsymbol{w}_0\|_2$ 均小于 $\Delta W_{\max}$.所以可以利用 $\Delta W_{\max}$ 来对 ΔW 进行有效的归一化处理.

针对 ΔC 而言,利用 3.3 节中求得 ΔC_1 来对 ΔC 进行归一化处理.对任意从权重空间 S_{region} 抽样得到的权重 $\boldsymbol{w}_i$,因为 $\boldsymbol{w}_1$ 为 $\boldsymbol{w}_0$ 到 S_{region} 的投影点,因此 ΔW_i 永远大于 ΔW_1.如果 $\Delta C_i > \Delta C_1$,那么意味着在 $\boldsymbol{w}_i$ 下面得到的新查询在评估模型式(1)下比在 $\boldsymbol{w}_1$ 下面得到的新查询差.所以在 $\boldsymbol{w}_i$ 下面得到的新查询不可能成为最优化的回答用户"为什么"问题的新查询,这类的 $\boldsymbol{w}_i$ 可以从候选集合中删除掉.这样仅仅需要考虑权重候选集合中那些使得 $\Delta C_i < \Delta C_1$ 的权重即可,因此可以利用 ΔC_1 来对 ΔC 进行有效的归一化处理.

因此,式(1)定义的评估模型通过归一化处理后如式(2)所示(在本文后面如果不作特殊说明,评估模型均为式(2)):

$$Eval(Q(k',\boldsymbol{w}')) = \lambda_w \frac{\Delta W}{\Delta W_{\max}} + \lambda_c \frac{\Delta C}{\Delta C_1}. \quad (2)$$

3.3.2 候选权重集合大小的确定

按照 3.1 节的讨论,候选权重通过抽样算法得到,因此为了提高抽样算法的效率,将权重抽样区域缩小到 S_{region},这样可以最大限度地保证抽样得到的权重有可能成为最优解.现在另外一个问题是抽样权重个数的确定,即需要从 S_{region} 抽样得到多少个权重,才能保证得到回答用户"为什么"问题的解最优.显然抽样得到的权重越多,得到最优解的可能性就越大.但是随着抽样得到权重的数目的增加,需要执行更多的渐进式 top-k 算法,从而增加了算法的运行时间.

为了解决这个问题,假设事件 A 为从权重空间 S_{region} 随机抽样一次得到的 $\boldsymbol{w}$ 为最优解,设 A 发生的概率 $P(A)=\alpha\%$,由于采用均匀抽样方法从 S_{region} 抽样得到 s 个候选权重,所以每一次的抽样均为独立事件.显然该抽样的过程符合 s 重 Bernoulli 实验.令 X 表示这 s 次抽样中事件 A 发生的次数.则:

$$X \sim B(s,\alpha\%).$$

假设经过 s 次抽样后,系统可以保证至少得到一个最优解概率大于 Pr,即

$$P(X \geqslant 1) = 1 - P(X=0) = 1 - (1-\alpha\%)^s \geqslant Pr, \quad (3)$$

从式(3)可以看出,抽样的数目完全取决于系统给定参数 α 和 Pr.当设定 α 为 0.2 且 Pr 为 0.9 时,表示通过 1 151 次抽样后,系统具有 90%可靠性可以得到一个近似最优解,该最优解至少比其他 99.8%的解要好.

3.3.3 渐进式 top-k 算法终止条件的改进

在 3.2 节具体讨论了对于给定的 $\boldsymbol{w}_i$,渐进式 top-k 算法执行多少步以后才能终止.接下来结合评估模型式(2),对渐进式 top-k 算法的终止条件进行进一步改进,提前终止那些没有可能得到更优新查询的渐进式 top-k 算法.

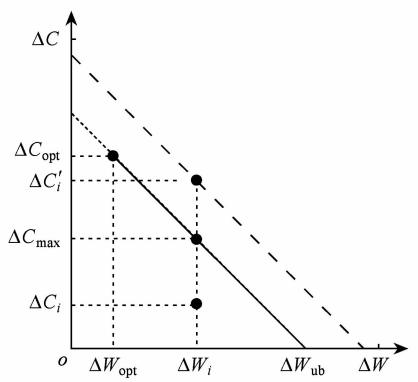


Fig. 2 Terminal condition of progressive top-k algorithm.

图 2 渐进式 top-k 算法终止条件

在算法 1 中,对于给定的权重向量 $\boldsymbol{w}_i$,求得参数 k 的上界 k_{ub},当渐进式 top-k 算法执行步骤达到上界值 k_{ub} 时,渐进式 top-k 算法终止执行.下面结合图 2 详细讨论下渐进式 top-k 算法的终止条件.图 2 中坐标轴为评估模型式(2)中的 2 个变量的值,其中横坐标表示 ΔW 的值,纵坐标表示 ΔC 的值.假设当前已经求得回答用户"为什么"问题的一个最优解为 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$(该最优解初始化为在权重 $\boldsymbol{w}_1$ 求得的新查询),ΔW_{opt} 表示 $\|\boldsymbol{w}_{\text{opt}} - \boldsymbol{w}_0\|_2$ 的值,ΔC_{opt} 表示最优解 Q_{opt} 和 Q_0 之间 top-k 结果的差异性.按照评估模型式(2),对 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$ 进行评估得到最优新查询的评估值为

$$E_{\min}=Eval(Q(k_{\text{opt}},\boldsymbol{w}_{\text{opt}}))=\lambda_w\frac{\Delta W_{\text{opt}}}{\Delta W_{\max}}+\lambda_c\frac{\Delta C_{\text{opt}}}{\Delta C_1}.$$

根据该评估值 $E_{\min}$,得到一个直线方程 l:

$$\frac{\lambda_w}{\Delta W_{\max}}\Delta W+\frac{\lambda_c}{\Delta C_1}\Delta C=E_{\min},$$

$\Delta W_{\text{ub}}=E_{\min}\Delta W_{\max}/\lambda_w$ 为该直线 l 和横坐标轴 ΔW 的交点.

给定候选权重集合 S 中的任意一个权重 $\boldsymbol{w}_i$,$\Delta W_i=\|\boldsymbol{w}_i-\boldsymbol{w}_0\|_2$ 为该权重和初始化权重 $\boldsymbol{w}_0$ 之间的差异,下面将从以下 2 种情况分别说明渐进式 top-k 算法的提前终止条件.

1) $\Delta W_i\geqslant\Delta W_{\text{ub}}$. 在这种情况下,因为 ΔC 永远不可能为负数,那么意味着在该 $\boldsymbol{w}_i$ 下面无论 k 值取多少,其得到的新查询 $Q(k,\boldsymbol{w}_i)$在评估模型式(2)下得到的值大于 $E_{\min}$,即该新查询永远不会比现有新查询 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$好. 这个结论的证明类似定理 1 的证明,这里不再累述. 所以在这种情况下,可以直接将该 $\boldsymbol{w}_i$ 删除掉,不需要对其执行渐进式 top-k 算法,从而减少了算法的执行开销.

2) $\Delta W_i<\Delta W_{\text{ub}}$. 如图 2 所示,将 ΔW_i 带入直线方程 l 可以计算出该 $\boldsymbol{w}_i$ 下 ΔC 的上界值 $\Delta C_{\max}$,即在该 $\boldsymbol{w}_i$ 下得到新查询的 ΔC 的值大于其上界值 $\Delta C_{\max}$ 时,该新查询在评估模型式(2)下不会优于现有的新查询 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$.

在渐进式 top-k 算法中算法每执行一步,结合 ΔC 的计算公式,可以得到每一步算法执行结束后,该 $\boldsymbol{w}_i$ 的 ΔC 都存在一个下界值(表示为 ΔC_{lb}). 在该下界下面,渐进式 top-k 算法的终止条件为 $\Delta C_{\text{lb}}>\Delta C_{\max}$. 综合考虑算法 1 中给出的渐进式 top-$k$ 算法终止条件,新的终止条件分为以下 2 种情况:

1) 当用户期望的对象 m 还没有出现的时候且已有 $\Delta C_{\text{lb}}\geqslant\Delta C_{\max}$,在这种情况下,将来求得的新查询的 ΔC 为图 2 中所示的 $\Delta C_i'$,该新查询在评估模型式(2)下绝对不会优于目前已有的新查询 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$. 在原有的算法 1 中,渐进式 top-$k$ 算法至少要执行到 m 出现,因此 ΔC_{lb}的引入可以提前终止渐进式 top-k 算法,提高算法的执行效率.

2) 当用户期望的对象 m 已经出现且 $\Delta C_{\text{lb}}<\Delta C_{\max}$时,继续执行渐进式 top-$k$ 算法求解 $\boldsymbol{w}_i$ 下最优的 k 值. 在这种情况下,渐进式 top-k 算法每执行一步,需要测试其终止条件:1)$k>k_{\text{ub}}$或者条件;2)$\Delta C_{\text{lb}}>\Delta C_{\max}$. 当渐进式 top-$k$ 算法满足其中的一个终止条件而终止时,将新得到新查询与目前已有的最优查询 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$进行评估分析,选择出最优的新查询去更新 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$.

综上所述,在评估模型式(2)对最优解的约束下,渐进式 top-k 算法的执行效率得到了很大的提升.

3.3.4 基于用户反馈的 top-k 告警修正算法

基于用户反馈的 top-k 告警修正算法,首先使用算法 2 求出的抽样空间 S_{region}和 $\boldsymbol{w}_0$ 在该空间上面的投影 $\boldsymbol{w}_1$,计算在 $\boldsymbol{w}_1$ 下的解并作为当前最优解 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$;然后在抽样空间抽样得到 s 个候选权重;针对每一个候选权重,执行渐进式 top-k 算法,并根据 3.3.3 节的讨论来设置渐进式 top-k 算法的终止条件. 最后将 $Q_{\text{opt}}(k_{\text{opt}},\boldsymbol{w}_{\text{opt}})$返回给用户作为本问题的最终最优解. 由于算法核心部分在本文的第 3 节已经详细讨论,因此这里不再对该算法进行详细解释.

下面对算法时间复杂度进行简单的分析. 该算法的时间复杂度主要由以下 2 个方面构成:首先是求 $\boldsymbol{w}_0$ 到 S_{region}的投影点,在这里采用文献[11]中方法进行求解. 该方法的时间复杂度为 $O(n^4)$,其中 n 为投影空间多面体的凸点个数. 在本文所研究的问题中,投影空间 $\Gamma=\{(\boldsymbol{m}-\boldsymbol{p})\cdot\boldsymbol{w}>0,w[i]\in[0,1],\sum_{i=1}^{d}w[i]=1\}$使得 n 非常小,因此这部分时间耗费可以忽略不计. 其次就是执行渐进式 top-k 算法的时间耗费. 通过文献[9]可以得知渐进 top-k 算法的时间复杂度为 $k+|skyline(D)|$,因此在候选权重集合 S 上,整个算法的时间复杂度为 $|S|\cdot(k+|skyline(D)|)$,其中 $|S|$为候选权重集合的规模.

4 实验与结果分析

本节对本文提出算法的执行效率进行实验分析. 因为本文提出的查询修改问题目前还没有针对这方面的研究,因此本节的性能评估主要针对本文提出的算法. 为了针对不同用户的需求进行评估,本节构建了 3 类不同的用户:

1) 容忍修改权重的用户(tolerate modify weight, TMW). 表示用户相对初始化查询结果而言,更能够容忍对权重 $\boldsymbol{w}$ 的修改.

2) 容忍修改结果的用户(tolerate modify results, TMR). 表示用户相对于修改初始化查询权重,更能够容忍对初始化 top-k 结果的修改.

3) 不介意的用户(never mind, NM). 表示用户对修改初始化查询的权重和结果都能容忍.

针对这 3 类不同用户,对于本文提出的评估模型式(2)给出了不同的参数. 针对 TMW 用户将 λ_c

和 λ_w 分别设置为 0.9 和 0.1;针对 TMR 用户将 λ_c 和 λ_w 分别设置为 0.1 和 0.9;针对 NM 用户将 λ_c 和 λ_w 分别设置为 0.5 和 0.5.

所有的实验代码都是通过 Java 实现并在 JDK 1.7 下面编译,程序在台式机上运行,该台式机安装 Ubuntu Linux 操作系统,CPU 为 intel Core-2 双核处理器,内存 2 GB. 实验所用的数据通过经典的数据生成算法产生[12]. 由于均匀分布数据集合相关数据集在 skyline 和 top-k 上面的性能表现差不多,因此在本节主要测试本文所提出的算法在均匀分布数据集和反相关数据集上的性能. 表 2 给出了产生模拟数据的参数,在本节后面的 4 组实验中,黑体表示默认参数,通过改变每个参数不同的取值来进行测试. 对于抽样的大小设置 3.3.2 节中参数 α 的默认值为 0.2,Pr 的默认值为 0.9. 给定一个初始化查询 $Q_0(k_0,\boldsymbol{w}_0)$,设置排在最后一位的对象为用户"为什么"问题里面的可比较对象 p,用户期望的对象 m 从 $D-Q_0$ 中随机产生但是必须满足 m 和 p 之间不存在相互支配的关系. 在这个前提下,通过测试程序的运行时间来衡量本文算法的有效性.

Table 2 Experimental parameters setting

表 2 实验参数设置

Parameters	Value
$10^{-3}\times$Data Size	1,**10**,50,100
Dimensionality	2,**3**,4,5
Origin k_0	5,**10**,15,20

第 1 组实验在数据维度为 3 的前提下,测试在不同数据规模下本文算法的运行时间. 如图 3 所示,本组实验是在 4 个不同数据规模和 3 个不同容忍度模型下完成的. 通过图 3 可以看出,本文算法的时间耗费随着数据规模的增长呈现出一种线性增长的关系. 在 TMR 模型下算法运行所耗费的时间最少,主要是因为在 TMR 模型中相对于初始化查询偏好的变化,用户更加能够容忍对初始化查询结果的修改,这样造成了图 2 中 ΔW_{ub} 与初始化最优新查询 Q_{opt} 的 ΔW_{opt} 之间的距离比较小,因此抽样的产生大部分权重在执行渐进式 top-k 算法之前会被删除掉.

第 2 组实验在数据规模为 1×10^4 的前提下,测试不在同数据维度下本文提出算法的运行时间. 实验结果如图 4 所示. 通过该图可以得出在数据规模保持不变的情况下,在均匀数据集上算法的时间耗费几乎保持不变(如图 4(a)所示),但是在反相关数据集上算法的时间耗费随着数据维度的增加呈现出

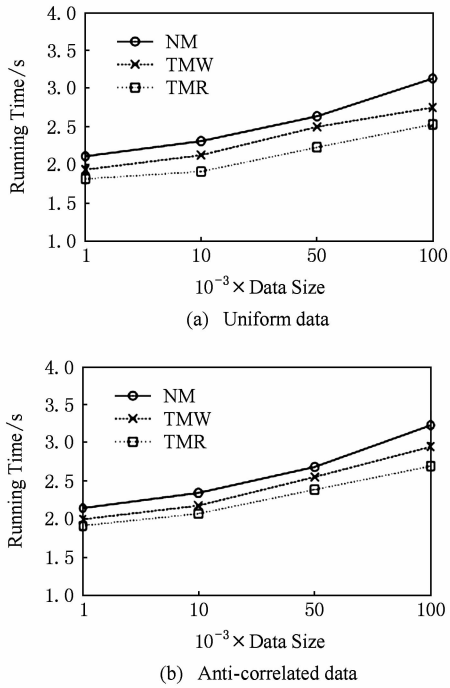


Fig. 3 Performance under different data size.

图 3 不同数据规模下程序的性能

比较快的增长趋势(如图 4(b)所示). 究其原因主要是因为在反相关数据集上执行渐进式 top-k 算法需要耗费比较多的时间,渐进式 top-k 算法在反相关数据集上随着数据维度的增加 top-k 候选集合规模呈现出较大的增长趋势.

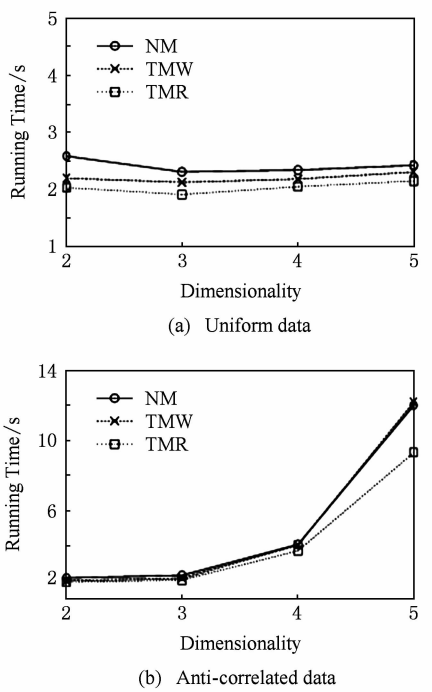
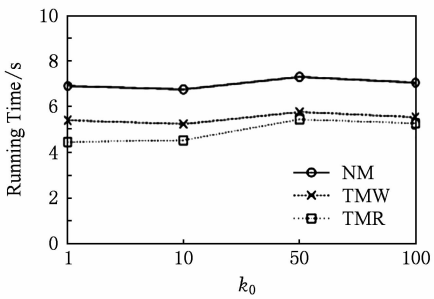


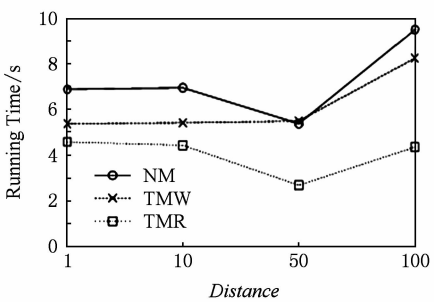
Fig. 4 Performance under different dimension.

图 4 不同数据维度下程序的性能

第 3 组实验是在数据规模为 1×10^4 且数据维度为 3 的前提下,测试在不同的初始化查询和用户期望对象下本文算法的运行时间. 通过实验发现本文算法的运行时间和 k_0 以及 m 之间关系不大. 因此在这里只给出了在反相关数据集上面程序的运行效率. 图 5(a)为变化 k_0 时程序的运行时间. 由于可比较对象 p 为初始化查询返回的最后一个对象,因此变化 k_0 相当于变化可比较对象 p. 同时设置对象 p 和用户期望的对象 m 在初始化查询下的排序距离不小于 10. 通过图 5(a)可以看出本文算法的运行时间和 k_0 的设置没有较强的关联性. 图 5(b)在固定 $k_0=10$ 的前提下,通过变化用户期望的对象 m 来测试本文算法效率. 其中 *Distance* 表示在初始化查询下对象 p 的排序位置和对象 m 的排序位置之间的最小距离. 可以看出用户期望对象 m 的选择对本文算法运行时间是有一定的影响,但是该影响仅仅限于对象 m 本身,与其在初始化查询下与对象 p 的距离关系不大.



(a) Uniform data

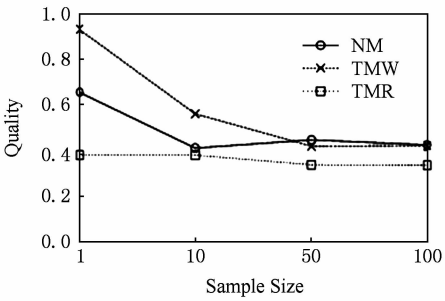


(b) Anti-correlated data

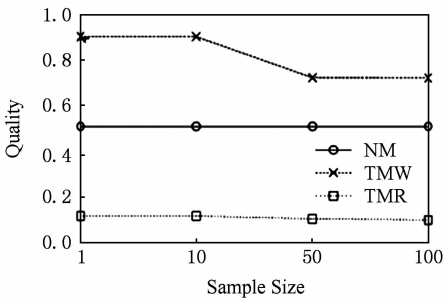
Fig. 5 Performance under different initial parameters.

图 5 不同初始化查询参数下程序的性能

第 4 组实验在数据规模为 1×10^4 且数据维度为 3 的前提下,测试抽样精度对“为什么”问题解的质量的影响. 通过 3.3.2 节的分析得知,抽样精度受参数 α 和参数 Pr 控制. 按照抽样精度的计算式(3)可以得知 α 越小 Pr 越大抽样精度越高. 因此在这里不再分别测试参数 α 和参数 Pr 变化时,得到新查询的质量. 图 6(a)给出了不同抽样精度下得到的新查询质量,可以看出随着抽样权重数目的增加,得到的新查询的质量越来越好. 同时在图 6(b)中新查询的质量并没有随着抽样数目的增加而增加,造成这种现象的原因是因为该新查询时在权重 $\boldsymbol{w}_0$ 的投影点 $\boldsymbol{w}_1$ 处取得的,因此无论抽样精度怎么变化,其得到最优新查询的质量保持不变.



(a) Uniform data



(b) Anti-correlated data

Fig. 6 Quality under different sampling accuracy.

图 6 不同抽样精度下新查询的质量

5 总 结

针对用户对系统返回的 top-k 结果的疑问,本文提出了一种基于用户反馈的 top-k 参数修正方法. 该方法通过从权重子空间抽样得到候选权重来重新定义新查询. 首先定义了用来衡量初始化 top-k 查询变化的评估模型函数. 基于该评估模型函数,对抽样方法的抽样空间进行了进一步的细粒度刻画,从而使得抽样空间更加精确. 最后在给定的候选权重上,通过执行渐进式 top-k 算法来得到新的最优化查询. 实验结果表明本文所提出的算法具有较好的执行效率.

参 考 文 献

[1] Tsaparas P, Palpanas T, Kotidis Y, et al. Ranked Join Indices[C]//Proc of the 19th Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2003: 277-288

[2] Chapman A, Jagadish H. Why not? [C] //Proc of the 2009 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2009: 523-534

[3] Motro A. FLEX: A tolerant and cooperative user interface to databases [J]. IEEE Trans on Knowledge and Data Engineering, 1990, 2(2): 231-246

[4] Huang Jiansheng, Chen Ting, Doan A, et al. On the provenance of non-answers to queries over extracted data [J]. The Proceedings of the VLDB Endowment (PVLDB), 2008: 1(1): 736-747

[5] Herschel M, Mauricio A, Hernández. Explaining missing answers to SPJUA queries [J]. The Proceedings of the VLDB Endowment (PVLDB), 2010: 3(1): 185-196

[6] Tran Q, Chan C. How to ConQueR why-not questions [C] //Proc of the 2010 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 15-26

[7] Vlachos A, Doulkeridis C, Kotidis Y, et al. Reverse top-k queries [C] //Proc of the 26th Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2010: 365-376

[8] He Z, Lo E. Answering Why-not Questions on Top-k Queries [C] //Proc of the 28th Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2012: 750-761

[9] Chen Lei, Zou Lei. Dominant Graph: An Efficient Index Structure to Answer Top-k Queries [C] //Proc of the 24th Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2008: 536-545

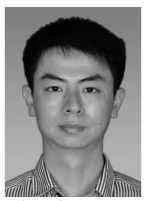
[10] Gan Liang, Jin Xin, Jia Yan, et al. Gridded dominant graph: A more efficient index structure for top-k queries based on reverse dominant point set [J]. Journal of Computer Research and Development, 2010, 47(10): 1771-1784 (in Chinese)
(甘亮, 金鑫, 贾焰, 等. GDG: 一种基于逆支配点集的 top-k 高效查询索引方法[J]. 计算机研究与发展, 2010, 47(10): 1771-1784)

[11] Golubitsky O, Mazalov V, Watt S M. An algorithm to compute the distance from a point to a simplex [J]. ACM Communications in Computer Algebra, 2012, 46(180): 57

[12] Börzsönyi S, Kossmann D, Stocker K. The Skyline Operator [C] //Proc of the 17th Int Conf on Data Engineering (ICDE'2001). Piscataway, NJ: IEEE, 2001: 421-430

Zhang Jianfeng, born in 1984. Received his master degree from National University of Defense Technology in 2009. Currently PhD candidate at National University of Defense Technology. His main research interests include database, network security and data mining.

Han Weihong, born in 1973. Received her PhD degree from National University of Defense Technology. Currently professor. Her main research interests include network & information security, database and data mining (hanweihong@gmail.com).



Fan Hua, born in 1984. Received his master degree in computer science from National University of Defense Technology in 2009. Currently PhD candidate at National University of Defense Technology. His main research interests include stream data management and Sensor network (huafan@nudt.edu.cn).

Zou Peng, born in 1957. Received his PhD degree from National University of Defense Technology. Professor and PhD supervisor at National University of Defense Technology. His main research interests include network & information security, distributed computing (zpeng@nudt.edu.cn).

Jia Yan, born in 1960. Received the PhD (AI and database) degree from National University of Defense Technology in 2001. She is a professor and PhD supervisor at National University of Defense Technology. Her main research interests include middle ware technology, massive database system, information security and data mining (jiayanjy@vip.sina.com).