ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, Supplement, December 2009, pp.286−297

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

一种关系数据库关键词检索相关反馈方法*

彭朝晖[1+], 崔立真[1], 王 珊[2,3], 张 俊[2,4], 王长亮[5]

[1](山东大学 计算机科学与技术学院,山东 济南 250101)

[2](数据工程与知识工程教育部重点实验室(中国人民大学),北京 100872)

[3](中国人民大学 信息学院,北京 100872)

[4](大连海事大学 计算机科学与技术学院,辽宁 大连 116026)

[5](香港科技大学,香港)

Method of Relevance Feedback in Keyword Search over Relational Databases

PENG Zhao-Hui[1+], CUI Li-Zhen[1], WANG Shan[2,3], ZHANG Jun[2,4], WANG Chang-Liang[5]

[1](School of Computer Science and Technology, Shandong University, Ji'nan 250101, China)

[2](Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education (Renmin University of China), Beijing 100872, China)

[3](School of Information, Renmin University of China, Beijing 100872, China)

[4](School of Computer Science and Technology, Dalian Maritime University, Dalian 116026, China)

[5](Hong Kong University of Science and Technology, Hong Kong, China)

+ Corresponding author: E-mail: pzh@sdu.edu.cn

Peng ZH, Cui LZ, Wang S, Zhang J, Wang CL. Method of relevance feedback in keyword search over relational databases. *Journal of Software*, 2009,20(Suppl.):286−297. http://www.jos.org.cn/1000-9825/09033.htm

Abstract: In keyword search over relational databases (KSORD), retrieval of user's initial query is often unsatisfying. User has to reformulate his query and execute the new query, which costs much time and effort. In this paper, a method of automatically reformulating user queries by relevance feedback is introduced. The method adopts a ranking method based on vector space model to rank KSORD results. Based on the results of user feedback or pseudo feedback, it computes expansion terms based on probability and reformulates the new query using query expansion. Experimental results verify that after KSORD systems executing the new query, more relevant results are presented to user.

Key words: relevance feedback; relational database; information retrieval; user feedback; pseudo feedback

摘 要: 在关系数据库关键词检索(KSORD)中,用户的检索往往不能一次成功,有时需要多次重构查询(找到一组新关键词)来进行检索,但是查询的重构往往要花费用户大量的时间和精力.针对 KSORD 的结果,提出了一种相关反馈方法来自动重构查询.该方法选用基于向量空间模型的打分机制对 KSORD 结果打分,根据用户反馈或伪反馈的结果信息,采用基于概率的方法计算扩展用的语词,以查询扩展的方法自动重构查询进行再次检索.实

* Supported by the National Natural Science Foundation of China under Grant Nos.60473069, 60496325 (国家自然科学基金); the China Postdoctoral Science Foundation under Grant No.200904501193 (中国博士后科学基金)

Received 2009-05-03; Accepted 2009-09-30

验结果表明,这种方法能够为用户提供更多的相关结果.

关键词: 相关反馈;关系数据库;信息检索;用户反馈;伪反馈

关系数据库关键词检索(keyword search over relational databases,简称KSORD)使得普通用户能够使用关键词来检索关系数据库,而不需要了解数据库模式和SQL语言[1].目前关于KSORD已经有许多研究工作,包括DBXplore[2],DISCOVER[3],IR-Style[4],BANKS[5,6],SEEKER[7],DETECTOR[8,9],SPARK[10]等.KSORD系统往往产生大量的检索结果,尽管系统提供排序(rank)机制,但是排序的规则并不一定符合当前查询的需要,往往结果列表中排在前面的很多结果都不是用户所需要的,用户需要花费较多的时间重构查询,再次检索[1].

相关反馈是信息检索(IR)中一种常用的查询重构技术,系统向用户返回一组文献,用户标出其中的相关文献,然后系统根据用户的相关性判断,自动重构查询来做二次检索,这种相关反馈形式又可称为用户反馈.相关反馈的目标是希望新的查询能够将相关的文献排在结果列表的前面[11,12].伪反馈是相关反馈的另一种形式[11],在伪反馈中,系统假定用户首次检索返回的文献列表中排在前面的若干文献是相关的,然后根据这些文献来重构查询.伪反馈的优势在于不需要用户参与,因为用户往往期望用尽可能简单的方式去检索信息,而不愿意去做诸如标注相关文献等额外的工作.

目前KSORD中对查询重构和相关反馈的研究非常少见.针对KSORD结果的特点,本文提出一种相关反馈方法.该方法首先选用基于向量空间模型的打分机制对KSORD结果打分,然后基于用户反馈或伪反馈的结果,采用一种基于概率的方法计算查询扩展所用的语词,以查询扩展的方式自动重构查询.实验结果表明,这种相关反馈方法能够在再次检索产生的结果列表中给用户提供更多的相关结果.

本文第1节介绍所要用到的基本概念.第2节介绍所提出的相关反馈方法.第3节介绍详细的实现算法.第4节给出实验结果.第5节和第6节分别是相关工作和全文总结.

1 基本概念

为了便于描述问题,我们首先定义本文所要用到的基本概念.

定义1(数据图). 一个关系数据库的数据可以被表示为一个无向加权的数据图$G(V,E)$,V是节点集,E是边集.其中,对于数据库中的每个元组t,G有一个相对应的节点$u_t \in V$.本文将数据库中的元组和G中的节点不加区别;对于数据库中的每一对元组s和t,如果s和t之间存在主外码引用关系,G中就包含一条无向边$\langle u_s,u_t \rangle \in E$.$G$中每个节点和边被预先按照一定规则赋予一个权值[5].

定义2(关键词查询). 用户的一次输入被定义为一个关键词查询Q,由一组关键词$k_1,\ldots,k_n$组成($n \geq 1$),表示为$Q(k_1,\ldots,k_n)$.

如果一个节点包含关键词作为其属性值的一部分,则称该节点和关键词是相关的,并称该节点为关键词节点.通常,KSORD系统查询执行过程的第1步是利用RDBMS提供的全文索引(或KSORD系统自己建立的全文索引),对每一个关键词k_i确定与其相关的关键词节点集合S_i.

定义3(结果树). 一次关键词查询的一个结果是一棵有根的加权树(元组连接树),它是数据图的一个子图,包含每个S_i中的至少1个节点,而且每个叶子节点必然是关键词节点.

KSORD系统按一定的打分方法为每棵结果树计算一个相关性分数,结果树按照相关性分数降序排列,系统返回用户所要求的top-k结果树.本文介绍和比较了基于VSM和基于树结构的两种有代表性的打分方法.

例如,图1是DBLP[13]数据库的模式,给定关键词查询*Hristidis Databases*,KSORD系统从DBLP的数

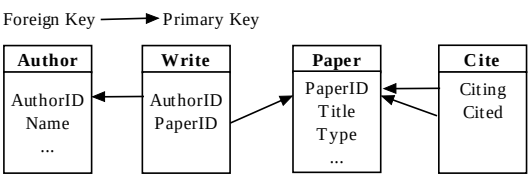

Fig.1 DBLP database schema

图1 DBLP数据库的模式

据图中检索出top-*k*结果树,每棵结果树是DBLP数据图的一个子图.图 2 给出了其中一棵结果树的示例,其含义是Hristidis写了一篇论文,论文题目是“Efficient IR-Style Keyword Search over Relation **Databases**”.

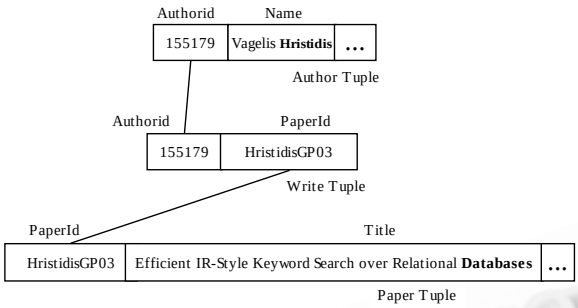

Fig.2 A result tree of DBLP database

图 2 DBLP 数据库的结果树示例

2 相关反馈方法

2.1 框 架

本文采用用户反馈和伪反馈两种方法实现KSORD中的相关反馈,其框架如图3所示.我们实现了一种基于向量空间模型(vector space model,简称VSM)的结果打分方法,当用户提交查询时,可以选择使用VSM的打分方法,也可以使用 KSORD 本身的打分方法.对于系统首次检索返回的结果,如果是用户反馈,则用户需要对最靠前的若干结果(如20个结果)进行相关性判断,给出相关结果的集合;如果是伪反馈,则不需要用户的参与,系统假定首次检索回来的前面几个结果(如前 5 个结果)是相关的.然后系统根据反馈的相关结果集合,计算扩展用的语词,重构查询,使用新查询进行再次检索,将检索结果再次呈现给用户.

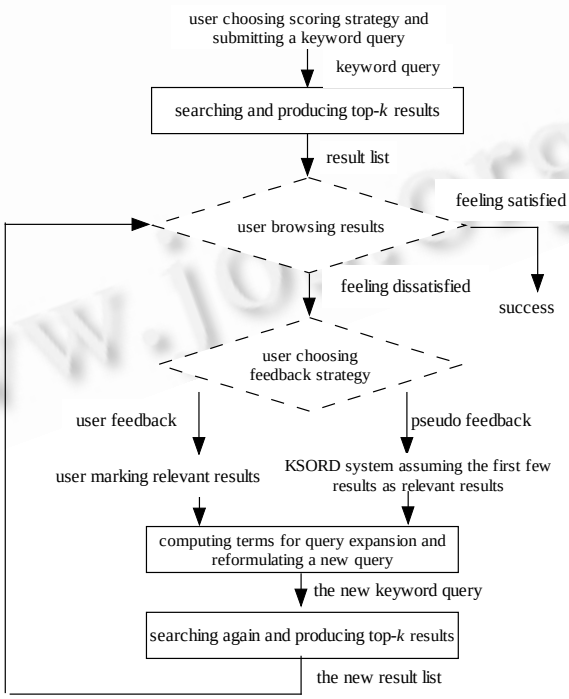

Fig.3 Framework of relevance feedback

图 3 相关反馈框架

2.2 基于VSM的结果打分

我们提出的相关反馈方法的基本思想是计算出相关结果中出现频率较高的语词,作为查询扩展的语词使用.这种查询扩展的查询重构方法在 IR 中大多用于基于 VSM 的打分机制的信息检索系统,因此我们首先给出KSORD中的基于VSM的结果打分方法,然后在此基础上讨论如何对KSORD做查询扩展.考虑KSORD结果的特点,我们选用了文献[4]中给出的针对结果树的基于 VSM 的打分方法,实验结果表明,这种打分方法确实提高相关反馈的效果.

这种打分机制是把关系的每个列当作一个文档集,把列在每个元组上的分量作为一个文档.

假设Q表示用户的查询,$tuple$表示结果树的一个节点,a_i表示这个元组的第i个属性,T表示整棵结果树.我们先对结果树的每个节点,计算出这个节点和查询Q之间的相关性分数.一个节点和查询间的分数等于它的每个属性和查询间的分数之和,即

$$Score(tuple,Q)=\sum_{a_i\in tuple} Score(a_i,Q).$$

整棵结果树的分数等于所有节点的分数和除以树的节点个数:

$$Score(T,Q)=\frac{\sum Score(tuple,Q)}{Size(T)}.$$

对于一个属性和查询间的相关性分数,则使用下式计算.该式是IR中一种基于VSM的相关性分数计算方法,其中使用TF-IDF公式的一种变形来计算语词的权值[12].

$$Score(a_i,Q)=\sum_{w\in Q\cap a_i}\frac{1+\ln(1+\ln(tf))}{(1-s)+s\dfrac{dl}{avdl}}\cdot\ln\frac{N+1}{df}.$$

在上式中,对于每一个既出现在查询Q,又出现在属性a_i中的词w,tf表示w在当前节点的a_i属性中出现的次数,df表示在a_i所属的关系上有多少个元组的a_i属性出现了w,dl是当前节点的a_i属性的内容的长度,$avdl$是a_i所属的关系中a_i属性内容的平均长度,s是一个常数,一般取经验值 0.2 左右,N是a_i所属的关系所包含的元组个数.从这个公式可以知道,只有关键词节点才对结果树的分数有贡献.

举例来说,在 DBLP 数据集上,计算图 2 中结果树的分数,得到的值为 5.81.

2.3 计算扩展语词

我们首先从直观上考虑,从反馈回来的结果中统计出现频率最高的那些词,使用它们作为扩展语词,是一个可行的做法.在这个直观设想基础上,为了使反馈的鲁棒性更好,本文选用一种基于概率的计算扩展语词的方法[14],下面介绍其基本思想.

令$D=(d_1,d_2,\ldots,d_l)$表示反馈的相关结果集合,d_i表示一棵结果树.我们通过下式得到扩展用词:

$$w=\arg\max_w p(w\mid D).$$

上式表示在反馈的相关结果集为 D 的条件下,选取被推断出来的概率最高的词 w 来进行查询扩展.根据贝叶斯公式,上式可变为

$$w=\arg\max_w\frac{p(D\mid w)\cdot p(w)}{P(D)}.$$

$P(D)$是一个常数,对于每个 w 都是一样的,可以忽略.所以我们又得到:

$$w=\arg\max_w p(D\mid w)\cdot p(w).$$

根据 KSORD 结果树的产生过程,我们可以假设每个结果树是否和查询 Q 相关是互相独立的事件,此时可得到:

$$w=\arg\max_w\prod_{i=1}^{l}p(d_i\mid w)\cdot p(w).$$

我们还可以假设,在一棵结果树d_i中,每个词t_j和t_k间的出现与否以及位置都是独立的,此时我们可以进一步简化模型为

$$w=\arg\max_{w}\prod_{i=1}^{l}\prod_{t\in d_i}p(t\mid w)\cdot p(w)\ .$$

对目标函数取对数不会影响目标函数的结果,因此,最后我们得到如下计算式:

$$w=\arg\max_{w}\sum_{i=1}^{l}\sum_{t\in d_i}\log(p(t\mid w))+\log(p(w))\ .$$

从上式可以看出,最后我们需要实际计算的只有 $p(w)$和 $p(t|w)$两个值.但是这两个值需要对每一个建立了全文索引的列分别计算,这是因为只有建立了全文索引的列才能被检索,其中的语词才能作为关键词使用,而我们的模型又是把一个列作为一个文档集.我们在每个建立全文索引的列上分别求出函数值最大的语词作为候选语词,再从候选语词中取函数值最高的 w 作为最终查询扩展使用的语词.根据需要,也可以取函数值最高的前 k 个词作为扩展语词使用.

对于 $p(w)$我们可以用 w 在对应关系上出现的元组数 df 除以整个关系的元组数 N 来估计,即 $p(w)=\frac{df_w}{N}$,而 $p(t|w)$可以用 w 和 t 两个词共同出现的的次数除以 w 出现的次数来估计,即 $p(t\mid w)=\frac{p(t,w)}{p(w)}=\frac{df_{t,w}}{df_w}$.注意到有时 t 从来都没有与 w 一起出现过,此时不能简单地把概率值赋为 0.当 t, w 的共同出现次数为 0 时,设整个列上所有出现过的不同的词的总个数为 NUM,采用 $\frac{1}{NUM}$ 作为 $p(t|w)$的估计值.这个估计值的隐含意义就是在没有任何其他信息的条件下,给定词 w,出现词 t 的概率等于均匀分布下随机从 NUM 个不同的词中挑出词 t 的概率.

举例来说,对于查询 *Hristidis Databases*,如果反馈结果集 D 只含有图 2 中的结果树这一个元素,则根据目标函数计算后,除了原查询关键词 *Hristidis* 和 *Databases* 以外,得分最高的是语词 *Keyword*,对应的目标函数的值为 −28.16.这样,重构而得的新查询为 *Hristidis Databases Keyword*.

3 算法实现

我们建立了两个倒排表的数据结构,用来支持基于 VSM 的结果打分和扩展语词的计算.对每一个建立了全文索引的列,都分别建立这样的两个数据结构.

描述语词在列中总体信息的倒排词典 *invertedList*,其成员包括:

Hashtable⟨*String w*,*Integer df*⟩ *frequencyList*,标识语词 w 在该列的多少个元组分量中出现;

double avdl,标识该列的内容平均长度(含语词个数);

int N,标识该列所属关系的总元组数.

描述列中语词在列所属关系中出现情况的倒排索引 *ivertedIndex*,是一个 Hash 表:

Hashtable⟨*String w*,*TreeSet*⟨*Item*⟩⟩*invertedIndex*.

Item 类的成员包括:*rowid tid*,*double tf*,*double dl*.其中,*tid* 标识语词 w 在该列的哪些元组分量中出现,*tf* 标识 w 在该元组分量中的语词频率,*dl* 标识该元组分量的内容长度(所含语词个数).

另外,我们建立一个无用词(stopword)集合 *HashSet*⟨*String*⟩*stopList*,无用词是指“*the*”,“*a*”等无实际意义的词,检索中需要将这些词过滤掉.

在系统启动之后,首先采用算法 1 和算法 2 对建立了全文索引的各个列创建倒排词典和倒排索引.在内存中建立这样两个数据结构,避免了对数据库的频繁操作,能够确保算法 3 和算法 4 有较高的执行效率.

算法 1. 创建倒排词典(*buildInvertedList* 算法).

输入:列 *col*,*col* 所在的关系 *tab*,对正文作规范化的正则表达式(regular expression)*Regex*.

输出:*col* 的倒排词典 *invertedList*.

Begin

01 初始化 *invertedList*;

02 使用 SQL 把关系 *tab* 的列 *col* 的全部内容读取到 *rs* 中;

03 **while** *rs.next*()

04 把 *col* 列的当前元组分量用 *Regex* 对应的规则做规范化,去掉不在 *Regex* 正则表达式规定内的字符;

05 将规范化后的字符串处理成为一个词集合 *words*;

06 **for** each *w* in *words* **do**

07 如果 *w* 是 *stopList* 中的词,则跳过 *w*;

08 在 *frequencyList* 中查找 *w*,若 *w* 没出现过,则将⟨*w*,1⟩加入到 *frequencyList*,否则把 *w* 的对应值加 1;

09 将 *N* 赋值为 *rs* 所含元组分量的个数,将 *avdl* 赋值为每个元组分量的内容长度之和除以 *N*;

10 **return** *invertedList*;

End

算法 2. 创建倒排索引(*buildInvertedIndex* 算法).

输入:列 *col*, *col* 所在的关系 *tab*,对正文做规范化的正则表达式 *Regex*.

输出:*col* 的倒排索引 *invertedIndex*.

Begin

01 初始化 *invertedIndex*;

02 使用 SQL 把 *tab* 的列 *col* 的全部内容读取到 *rs* 中;

03 **while** *rs.next*()

04 把 *col* 列的当前元组分量用 *Regex* 对应的规则做规范化,去掉不在 *Regex* 正则表达式规定内的字符;

05 将规范化后的字符串处理成为一个词集合 *words*,计算分量的内容长度 *dl*;

06 **for** each *w* in *words* **do**

07 如果 *w* 是 *stopList* 中的词,则跳过 *w*;

08 计算 *w* 在当前元组分量中的语词频率 *tf*;

09 在*invertedIndex*中查找*w*,若*w*没出现过,则加入键*w*,设其值为仅含元素*item*的集合,*item*成员变量的值分别为当前元组ID,*tf*,*dl*,否则,将*item*添加到键*w*对应的值集合中;

10 **return** *invertedIndex*;

End

在系统检索过程中,采用算法 3 对检索到的结果打分,实验部分将证明这种基于 VSM 的打分方法比 KSORD 原有的打分方法更适合相关反馈的需要.

算法 3. 对结果使用 VSM 模型打分(Score 算法).

输入:结果树 *tree*.

输出:*tree* 的分数 *score*.

Begin

01 **for** each 关键词节点 *node* in *tree* **do**

02 **for** each *w* in *query* **do**

03 **if** *w* in *node* **then**

04 **for** *w* 所在的每个列 *col* **do**

05 查找 *col* 的 *InvertedList*,得到 *w* 在 *col* 上的 *df*,*col* 的 *avgdl*,以及 *col* 所属关系的元组总数 *N*;

06 查找 *col* 的 InvertedIndex,得到 *w* 在 *node* 的 *col* 分量上的 *dl* 和 *tf*;

07 使用 $Score(a_i,Q)$ 公式的分项部分(TF-IDF 公式的变形)计算 *w* 的权重;

08 把计算得到的值加入到 *sum* 中;

09 将 *sum* 除以树的节点个数 *Size*(*tree*),得到 *tree* 的分数 *score*;

10 **return** *score*;

End

将结果排序输出后,系统根据用户反馈或者伪反馈的结果计算出查询扩展所需的语词,将扩展语词和原关键词查询一起组成新的查询,提交给检索引擎再次检索,并将新的结果列表呈现给用户.算法 4 给出了在某一列上计算候选扩展语词的算法,对于建立全文索引的每个列,都执行算法 4,然后从生成的候选扩展语词中抽取函数值最高的若干个即可.

算法 4. 计算扩展用语词(*ComputeTerms* 算法).

输入:检索结果集合 *result*,反馈为相关的结果集合 *relevance*,需要扩展的语词个数 *k*,列 *col*.

输出:列 *col* 中可用来作查询扩展的候选语词及其函数值.

Begin

01 将 *relevance* 中 *col* 列的所有语词抽取出来形成候选词集合 *C*(不含无用词);

02 **for** each *w* in *C* **do**

03 查找列*col*的倒排词典*invertedList*,得到*w*在*col*的元组中的出现频率*df*,以及列*col*所属关系的总元组数*N*,计算*p*(*w*)=*df*/*N*;

04 **for** each *tree* in *relevance* **do**

05 **for** each *node* in *tree* **do**

06 若 *node* 不包含 *col* 列,则跳过此节点;

07 把 *node* 中 *col* 列的内容抽取出来,用 *Regex* 对应的规则做规范化,去掉不在 *Regex* 正则表达式规定内的字符,规范化后的字符串通过处理成为一个词的集合 *words*;

08 **for** each *t* in words **do**

09 查找列 *col* 的倒排索引 *invertedIndex*,得到 *w* 对应的元组集合 *CW*,以及 *t* 和 *w* 共同出现的集合 *CWT*;

10 若 *CWT* 为 *NULL*,则 *p*(*t*|*w*)=1/*Size*(*invertedIndex*),否则 *p*(*t*|*w*)=|*CWT*|/|*CW*|;

11 将 *log*(*p*(*t*|*w*))+*log*(*p*(*w*))加到 *w* 对应的 *sum* 中;

12 对 *C* 中每个 *w* 根据对应的 *sum* 值从大到小排序,取前 *k* 个词.注意前 *k* 个词不包含用户的查询关键词.

End

4 实验评估

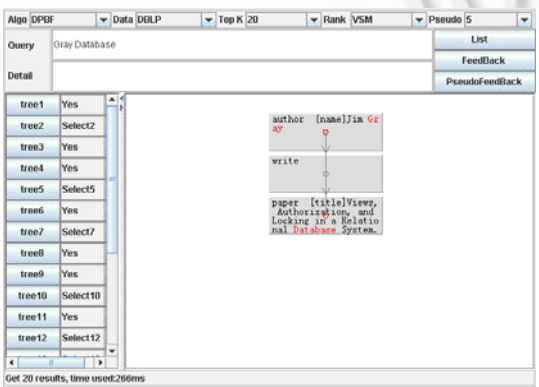

Fig.4 GUI of relevance feedback

图 4 相关反馈的图形用户界面

我们使用 Java 语言实现了相关反馈系统,其图形用户接口如图 4 所示.用户输入查询关键词,选择数据集,检索引擎,top-*k*,结果打分方法等参数,使用按钮“*List*”进行首次检索.若用户决定采用伪反馈重构查询,则点击按钮“*PseudoFeedBack*”,系统根据用户指定的参数 *pseudo* 的值取前 *pseudo* 个结果作为相关结果;若采用用户反馈,则用户需要阅读结果列表中靠前的若干结果,将相关的结果选择为“*Yes*”,然后点击按钮“*FeedBack*”.

我们的实验所用KSORD引擎选用DPBF算法[9],使用了BANKS系统[5]的打分机制.数据集采用DBLP数据库的全集[13],在关系*Author*的*Name*,*Paper*的*Title*和*Type*列上建立了全文索引.

4.1 实验方案

本文的研究重点是在KSORD中应用相关反馈技术来给用户提供更多的相关结果,所以实验中不涉及对系统效率的评估.在算法实现中,我们在内存中建立了倒排表,基于 VSM 的结果打分和计算扩展语词的算法主要都是内存操作,算法的效率很高.但是倒排表占用了一定的内存,从而对基于数据图的 KSORD 系统的执行效率产生了一定的影响,对此,只要适当增大机器内存,效率问题即可得到解决.

对于相关反馈效果的测试,我们从文献[7]实验所用的 20 个例子中随机抽取了 10 个作为测试用例,见表 1.文献[7]中的例子是用来测试 SEEKER 系统的查询效果的,并非专门为我们的相关反馈方法构建,用它来测试相关反馈的效果具有说服力.

Table 1 Test case of relevance feedback

表 1 相关反馈测试用例

Knuth algorithm	Codd relational	Dewitt database	Foster grid	Salton information retrieval
Dijkstra programming	Hartmanis NP	Jiawei han data mining	Stonebraker object	Bernstein concurrency

我们分别测试用户反馈和伪反馈的效果.对于同样的 top-k 值,如果二次检索比首次检索的结果列表中包含更多的相关结果,就说明我们的相关反馈是有效的,所以我们使用查准率来评估相关反馈的效果.对每项测试,我们将 10 个查询的查准率求平均值,查准率计算公式为 $Precision=\frac{|R|}{|K|}$.其中,$|R|$为相关结果的个数,由用户来判断,$|K|$即 top-$k$ 的 k 值.

对每一种反馈,我们分别测试 DPBF 和 VSM 打分机制下的平均查准率.对每一种打分机制,又分别测试 Top-10 和 Top-20 的情况,对每种情况,分别测试扩展语词个数为 1 和 2 的情形.

需要说明的是,对扩展两个语词的情况,在重构查询时,我们规定扩展出的两个语词之间是 OR 关系.例如,原有查询为 *Gray Database*,假设扩展出的两个语词为 *Transaction* 和 *Relational*,则新的查询为 *Gray AND Database AND*(transaction or relational).

4.2 用户反馈

表 2 和表 3 分别给出在 DPBF 和 VSM 的打分机制下,每个测试用例的用户反馈效果.表中的 List 和 List-VSM 分别表示在 DPBF 和 VSM 打分机制下首次检索的结果情况,Feed-1 和 Feed-2 分别表示扩展 1 个和 2 个语词后再次检索的效果.

表中的每一项测试值为结果列表中第 1 个相关结果和最后一个相关结果在列表中分别所处的位置,以及列表中符合条件的结果总数目,例如,5/9(3)表示第 1 个相关结果排在结果列表的第 5 位,最后一个相关结果排在第 9 位,符合条件的结果共有 3 个.

Table 2 User feedback using scoring strategy of DPBF

表 2 基于 DPBF 打分方法的用户反馈

No.	Query	Top-10			Top-20		
		List	Feed-1	Feed-2	List	Feed-1	Feed-2
1	Knuth algorithm	6/7 (2)	1/10 (10)	1/5 (5)	6/7 (2)	1/20 (19)	1/20 (9)
2	Dijkstra programming	1/9 (4)	1/10 (8)	1/10 (5)	1/9 (4)	1/19 (11)	1/15 (9)
3	Codd relational	5/10 (5)	1/10 (6)	1/10 (4)	5/20 (12)	1/20 (13)	1/20 (13)
4	Hartmanis NP	3/10 (5)	3/10 (8)	3/9 (7)	3/16 (8)	3/10 (8)	3/9 (7)
5	Dewitt database	1/10 (5)	1/10 (7)	1/10 (7)	1/20 (10)	1/20 (14)	1/20 (13)
6	Jiawei Han data mining	1/10 (5)	1/10 (7)	1/10 (5)	1/14 (7)	1/19 (13)	1/20 (11)
7	Foster grid	3/6 (4)	1/8 (5)	2/9 (7)	3/18 (7)	1/15 (8)	2/19 (12)
8	Stonebraker object	2/10(5)	1/6(6)	1/10(9)	2/19(7)	1/20(7)	1/20(12
9	Salton information retrieval	5/10(3)	1/10(8)	2/9(7)	5/20(8)	1/20(14)	1/20(14)
10	Bernstein concurrency	1/10(5)	1/9(5)	1/7(6)	1/20(11)	1/20(12)	1/20(12)

Table 3 User feedback using scoring strategy of VSM

表 3 基于 VSM 方法的用户反馈

No.	Query	Top-10			Top-20		
		List-VSM	Feed-1	Feed-2	List-VSM	Feed-1	Feed-2
1	Knuth algorithm	2/8 (2)	1/10 (10)	1/10 (10)	2/20 (3)	1/20 (20)	1/20 (20)
2	Dijkstra programming	1/2 (2)	1/9 (6)	1/5 (4)	1/14 (4)	1/20 (19)	1/19 (11)
3	Codd relational	1/9 (5)	1/10 (8)	1/9 (4)	1/20 (13)	1/20 (14)	1/19 (12)
4	Hartmanis NP	4/10 (4)	3/10 (8)	4/10 (4)	4/18 (9)	3/13 (10)	4/20 (7)
5	Dewitt database	1/3 (3)	1/5 (5)	1/10 (10)	1/19 (9)	1/20 (17)	1/20 (16)
6	Jiawei han data mining	1/10 (5)	1/10 (10)	1/10 (10)	1/20 (11)	1/20 (20)	1/20 (17)
7	Foster grid	2/10 (5)	1/9 (7)	1/10 (10)	2/16 (10)	1/20 (13)	1/18 (15)
8	Stonebraker object	3/9(5)	1/8 (8)	1/10 (10)	3/15 (10)	1/20 (13)	1/20 (16)
9	Salton information retrieval	5/8(2)	3/10 (8)	1/10 (8)	5/20 (8)	1/20 (15)	2/20 (14)
10	Bernstein concurrency	1/9(5)	1/10 (8)	1/10 (8)	1/20 (12)	1/20 (14)	1/20 (12)

将表 2 和表 3 的 10 个查询的查准率求平均值,得到了图 5.

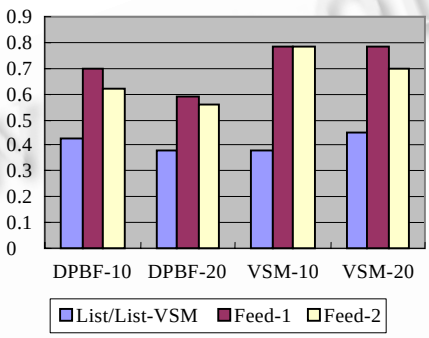

Fig.5 Effectiveness of user feedback

图 5 用户反馈的效果

从图 5 可以总结出如下结论:

1. 使用了用户反馈之后,查准率增长了很多,检索效果有了明显的改善.
2. Feed-2 的效果比 Feed-1 的效果略差,原因是 KSORD 在 DBLP 数据集上的检索结果所含的内容长度比较短,扩展过多的语词有时偏离了用户的查询要求.
3. 使用 VSM 的打分方法,在初次检索时,效果并不比 DPBF 打分机制的效果好,原因仍然是因为 KSORD 的结果的内容长度较短,VSM 方法没有充分发挥作用.但是用户反馈的效果,VSM 打分机制下要明显好于 DPBF 打分机制下.也就是说,VSM 的打分机制更适合于基于查询扩展的用户反馈.

4.3 伪反馈

表 4 和表 5 分别给出了 DPBF 和 VSM 的打分机制下,每个测试用例的伪反馈效果.系统使用前 5 个结果作为相关结果反馈.

将表 4 和表 5 的 10 个查询的查准率求平均值,得到了图 6.从图 6 可以看出,使用 VSM 的打分方法,伪反馈的效果要明显好于 DPBF 打分方法下的效果.

图 6 中可以看出,伪反馈不如用户反馈的平均查准率高,原因是 VSM 方法在首次检索时没有充分发挥作用,首次检索的效果不很理想,进而影响了伪反馈的效果.但是相对于用户反馈,伪反馈不需要用户参与,用户使用 KSORD 系统的效率更高.另外,用户还可以把伪反馈和用户反馈结合起来使用,多次重构查询,以尽快获得满意的结果列表.

Table 4 Pseudo feedback using scoring strategy of DPBF

表 4 基于 DPBF 打分方法的伪反馈

No.	Query	Top-10			Top-20		
		List	Feed-1	Feed-2	List	Feed-1	Feed-2
1	Knuth algorithm	6/7(2)	(0)	(0)	6/7(2)	(0)	(0)
2	Dijkstra programming	1/9(4)	3/5(2)	3/5(2)	1/9(4)	3/19(4)	3/19(4)
3	Codd relational	5/10(5)	1/9(6)	3/9(4)	5/20(12)	1/16(10)	3/20(8)
4	Hartmanis NP	3/10(5)	3/10(8)	3/8(6)	3/16(8)	3/17(11)	3/18(9)
5	Dewitt database	1/10(5)	1/10(6)	1/10(7)	1/20(10)	1/20(10)	1/20(14)
6	Jiawei Han data mining	1/10(5)	1/7(4)	1/6(5)	1/14(7)	1/17(8)	1/14(6)
7	Foster grid	3/6(4)	1/10(6)	1/7(6)	3/18(7)	1/17(8)	1/20(10)
8	Stonebraker object	2/10(5)	1/5(5)	1/10(10)	2/19(7)	1/20(8)	1/10(10)
9	Salton information retrieval	5/10(3)	5/8(3)	5/10(5)	5/20(8)	5/19(12)	5/19(10)
10	Bernstein concurrency	1/10(5)	1/9(5)	1/10(5)	1/20(11)	1/20(12)	1/20(12)

Table 5 Pseudo feedback using scoring strategy of VSM

表 5 基于 VSM 方法的伪反馈

No.	Query	Top-10			Top-20		
		List-VSM	Feed-1	Feed-2	List-VSM	Feed-1	Feed-2
1	Knuth algorithm	2/8(2)	(0)	(0)	2/20(3)	(0)	(0)
2	Dijkstra programming	1/2(2)	1/5(5)	1/6(4)	1/14(4)	1/5(5)	1/17(5)
3	Codd relational	1/9(5)	2/10(8)	3/9(5)	1/20(13)	2/17(11)	3/17(9)
4	Hartmanis NP	4/10(4)	3/10(8)	3/10(7)	4/18(9)	3/19(14)	3/20(13)
5	Dewitt database	1/3(3)	1/10(9)	1/10(9)	1/19(9)	1/20(14)	1/19(13)
6	Jiawei Han data mining	1/10(5)	2/9(5)	2/7(5)	1/20(11)	2/20(10)	2/20(12)
7	Foster grid	2/10(5)	3/9(5)	2/7(3)	2/16(10)	3/16(10)	2/19(10)
8	Stonebraker object	3/9(5)	1/10(10)	1/9(9)	3/15(10)	1/20(20)	1/17(13)
9	Salton information retrieval	5/8(2)	6/9(4)	6/10(5)	5/20(8)	6/20(12)	6/20(12)
10	Bernstein concurrency	1/9(5)	2/8(6)	1/8(6)	1/20(12)	2/20(11)	1/20(15)

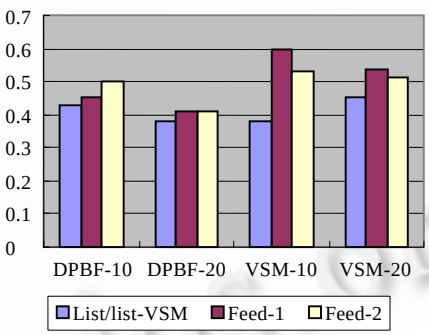

Fig.6 Effectiveness of pseudo feedback

图 6 伪反馈的效果

5 相关工作

不同的KSORD系统给出了不同的结果打分方法.IR-Style[4]中提出了基于VSM的打分方法,本文选用了这种方法.在最近的研究中,文献[15]在VSM的基础上,进一步细化了打分方法.BANKS[5]提出一种典型的基于结果树结构的打分方法,本文实验中的DPBF算法就是采用了这种打分机制.在BANKS中,结果树T的分数$Score(T)=(1-\lambda)Escore+\lambda Nscore$,其中,$Escore$是边的总分,$Nscore$是节点的总分,$\lambda$是控制因子.边的总分$Escore=1/(1+\Sigma_e Escore(e))$,其中,$Escore(e)=W(e)/W_{min}$,$W(e)$是边$e$的权值,其值由$e$的两个端点所属关系的密切程度(系统管理员指定)而定,关系越密切,$W(e)$的值越小;W_{min}是数据图中最小的边权值.节点的总分定义为结果树的根节点和所有叶节点(叶结点都是关键词节点)分数的平均值,每个节点的分数$Nscore(v)=N(v)/N_{max}$,其中$N(v)$是节点v的权值,通常定义为节点的度,值越大,说明节点越重要;N_{max}是数据图中最大的节点权值.正如本文的实验分析

所示,这种打分机制不适于基于查询扩展的相关反馈方法.

在IR领域,相关反馈的研究已经开展了数十年的时间,所采用的基本技术是查询扩展和语词重新加权[11].相关反馈最初的大部分工作是Rocchio在 60 年代晚期做的,Rocchio的早期工作随后被Ide扩展,他们的工作都是基于VSM[16].后来,在大量的文献开始研究在概率模型的基础上进行相关反馈.文献[17]在概率模型的基础上提出把语言模型(language model)运用到IR中,随后有大量的研究工作围绕语言模型展开.本文在计算查询扩展语词时所用方法和语言模型是不一样的,针对KSORD结果的特点,本文的方法比语言模型方法做了更多的假设,从而得到一个简化的模型,概率的计算简单,系统开销小,并且仍然得到较好的效果.

除了用户反馈和伪反馈之外,IR中还有一个值得关注的相关反馈方法是隐式反馈(implicit feedback).隐式反馈根据用户以往提交的查询以及用户以往对检索结果的操作情况,给用户提供个性化的结果列表[18].

KSORD 系统对相关反馈的研究非常少见.文献[19]支持用户标注出相关的结果,系统在数据图上检索结果树时根据用户的标注对某些路径进行裁减或给予更高的权值,这种相关反馈是一种结合 BANKS 的实现机制的用户反馈技术,但是作为一篇 DEMO 论文,文献[19]只是简单地提及这一想法,没有给出具体实现方法.文献[14]提出了 UFBP 方法,将用户反馈引入到 KSORD 中,并提出基于概率的方法计算扩展语词,这种方法为本文所采用,但是文献[14]没有考虑伪反馈和基于 VSM 的打分策略,而这两个方面是本文的重点.

6 结束语

本文提出一种适用于 KSORD 系统的相关反馈方法.该方法首先采用基于 VSM 的打分机制对 KSORD 结果打分,根据用户反馈或伪反馈的结果信息,选用一种基于概率的方法计算扩展语词,以查询扩展的方法实现查询重构.实验结果表明,这种相关反馈方法能够给用户提供更多的相关结果.在未来的工作中,我们考虑进一步针对 KSORD 结果的特点,结合数据库的模式和隐式反馈来改善方法的效果.

References:

[1] Wang S, Zhang KL. Searching databases with keywords. Journal of Computer Science and Technology, 2005,20(1):55−62.

[2] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases. In: Agrawal R, Dittrich K, Ngu AH, eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Press, 2002. 5−16.

[3] Hristidis V, Papakonstantinou Y. DISCOVER: Keyword search in relational databases. In: Bernstein PA, Ioannidis Y, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 670−681.

[4] Hristidis V, Gravano L, Papakonstantinou Y. Efficient IR-style keyword search over relational databases. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 850−861.

[5] Bhalotia G, Hulgeri A, Nakhe C, Chakrabarti S, Sudarshan S. Keyword searching and browsing in databases using BANKS. In: Agrawal R, Dittrich K, Ngu AH, eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Press, 2002. 431−440.

[6] Kacholia V, Pandit S, Chakrabarti S, Sudarshan S, Desai R, Karambelkar H. Bidirectional expansion for keyword search on graph databases. In: Böhm K, Jensen CS, Haas LM, Kersten ML, Larson P, Ooi BC, eds. Proc. of the 31st Int'l Conf. on Very Large Data Bases. Trondheim: ACM, 2005. 505−516.

[7] Wen JJ, Wang S. SEEKER: Keyword-Based information retrieval over relational databases. Journal of Software, 2005,16(7): 1270−1281 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1270.htm

[8] Cai HY, Yao JL, Wang S. DETECTOR: A universal on-line keyword search system over relational database. Journal of Computer Research and Development, 2007,44(1):119−125 (in Chinese with English abstract).

[9] Ding BL, Yu J, Wang S, Qin L, Zhang X, Lin XM. Finding Top-*k* min-cost connected trees in databases. In: Proc. of the 23rd Int'l Conf. on Data Engineering. IEEE Press, 2007. 836−845.

[10] Luo Y, Lin XM, Wang W, Zhou XF. SPARK: Top-*k* keyword query in relational databases. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Beijing: ACM, 2007. 115−126.

[11] Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Beijing: China Machine Press, 2004.

[12] Singhal A. Modern information retrieval: A brief overview. IEEE Data Engineering Bulletin, 2001,24(4):35−43.

[13] DBLP bibliography.2004. http://www.informatik.uni-trier.de/~ley/db/index.html

[14] Peng ZH, Zhang J, Wang S, Wang CL. Bring user feedback into keyword search over databases. In: Proc. of the 3rd Workshop on Electronic Government Technology and Application. 2009.

[15] Liu F, Yu C, Meng WY, Chowdhury A. Effective keyword search in relational databases. In: Chaudhuri S, Hristidis V, Polyzotis N, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Chicago: ACM, 2006. 563−574.

[16] Lundquist C, Grossman DA, Frieder O. Improving relevance feedback in the vector space model. In: Golshani F, Makki K, eds. Proc. of the 6th Int'l Conf. on Information and Knowledge Management. Las Vegas: ACM, 1997. 16−23.

[17] Ponte JM, Croft WB. A language modeling approach to information retrieval. In: Proc. of the 21st Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Melbourne: ACM, 1998. 275−281.

[18] Shen XH, Tan B, Zhai CX. Context sensitive information retrieval using implicit feedback. In: Baeza-Yates R, Ziviani N, Marchionini G, Moffat A, Tait J, eds. Proc. of the 28th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Salvador: ACM, 2005. 43−50.

[19] Aditya B, Chakrabarti S, Desai R, Hulgeri A, Karambelkar H, Nasre R, Parag, Sudarshan S. User interaction in the BANKS system: A demostration. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore: IEEE Press, 2003. 786−788.

附中文参考文献:

[7] 文继军,王珊.SEEKER:基于关键词的关系数据库信息检索.软件学报,2005,16(7):1270−1281. http://www.jos.org.cn/1000-9825/16/1270.htm

[8] 蔡宏艳,姚佳丽,王珊.DETECTOR:基于关系数据库通用的在线关键词查询系统.计算机研究与发展,2007,44(1):119−125.

彭朝晖(1978－),男,山东临清人,博士,讲师,主要研究领域为数据库信息检索.

张俊(1971－),男,博士,副教授,主要研究领域为数据库信息检索.

崔立真(1976－),男,博士,副教授,主要研究领域为软件与数据工程.

王长亮(1985－),男,博士生,主要研究领域为数据库信息检索.

王珊(1944－),女,教授,博士生导师,主要研究领域为高性能数据库新技术,内存数据库系统,数据库信息检索,数据仓库,BI技术.