纯 Peer to Peer 环境下有效的 Top-*k* 查询*

何盈捷+, 王 珊, 杜小勇

(中国人民大学 信息学院,北京 100872)

Efficient Top-*k* Query Processing in Pure Peer-to-Peer Network

HE Ying-Jie+, WANG Shan, DU Xiao-Yong

(Information School, Renmin University of China, Beijing 100872, China)

+ Corresponding author: Phn: +86-10-62519453, E-mail: heyj@ruc.edu.cn, http://www.ruc.edu.cn

Received 2003-12-02; Accepted 2004-04-01

He YJ, Wang S, Du XY. Efficient Top-*k* query processing in pure peer-to-peer network. *Journal of Software*, 2005,16(4):540−552. DOI: 10.1360/jos160540

Abstract: Most of the existing peer-to-peer (P2P) systems only support simple title-based search, and users cannot search the data based on their content. Top-*k* query is widely used in the search engine and gains great success. However, Processing top-*k* query in pure P2P network is very challenging because a P2P system is a dynamic and decentralized system. An efficient hierarchical top-*k* query processing algorithm based on histogram is proposed. First, a distributed query processing model for top-*k* query is proposed. It does top-*k* query in a hierarchical way. Ranking and merging of documents are distributed across the peers, which takes full advantage of the computing resource of the network. Next, a histogram is constructed for each peer according to the top *k* results returned by the peer, and used to estimate the possible upper bound of the score for the peer. By the histogram information, the most possible peers are selected to send the query, so as to greatly improve the search efficiency. Experimental results show that the top-*k* query improves the query effectiveness, and the histogram improves the query efficiency.

Key words: P2P network; top-*k* query; search; histogram

摘 要: 目前大多数的 Peer-to-Peer(P2P)系统只支持基于文件标识的搜索,用户不能根据文件的内容进行搜索.Top-*k* 查询被广泛地应用于搜索引擎中,获得了巨大的成功.可是,由于 P2P 系统是一个动态的、分散的系统,在纯的 P2P 环境下进行 top-*k* 查询是具有挑战性的.提出了一种基于直方图的分层 top-*k* 查询算法.首先,采用层次化的方法实现分布式的 top-*k* 查询,将结果的合并和排序分散到 P2P 网络中的各个节点上,充分利用了网络中的资源.其次,

* Supported by the National Natural Science Foundation of China under Grant No.604963205, 60473069 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA4Z3130 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2001CCA03000 (国家重点基础研究发展规划(973)); the Key Science-Technology Project of Beijing of China under Grant No.H030130040011 (北京市科技计划重大项目)

作者简介: 何盈捷(1977−),男,浙江诸暨人,博士,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,数据仓库,知识工程;杜小勇(1963−),男,教授,博士生导师,主要研究领域为高性能数据库,智能信息检索,知识工程.

根据节点返回的结果为节点构建直方图,利用直方图估计节点可能的分数上限,对节点进行选择,提高了查询效率.实验证明,top-*k* 查询提高了查询效果,而直方图则提高了查询效率.

关键词: P2P 网络;top-*k* 查询;搜索;直方图

中图法分类号: TP311 **文献标识码**: A

Peer-to-Peer(P2P),又称对等网络或对等计算,是当前研究的热点.目前大多数的 P2P 系统只支持基于文件标识的搜索[1−3],限制了 P2P 系统的应用.Top-*k* 查询在 Web 搜索引擎中被广泛地使用并获得了巨大的成功.例如 Google,每次查询,它都返回匹配度最高的 top 10 个 Web 页面.我们希望使用 P2P 系统也能像使用 Web 搜索引擎一样方便,当输入一个查询,返回 top *k* 个最匹配的结果.由于用户往往不关心所有返回的结果,而只关心很小一部分的结果,因此,在 P2P 中进行 top-*k* 查询是十分有用的,它能节约网络的带宽,同时也提高了用户的满意度.

由于 P2P 系统中不存在中心控制节点,网络中的节点也可以随意加入或退出系统,因此在 P2P 环境下进行 top-*k* 查询是十分具有挑战性的.本文讨论了如何在纯的 P2P 环境下进行有效的 top-*k* 查询,我们将查询的范围主要集中在非结构化文档上,给定一个查询,可能是几个关键字,一个短语,一个句子,或者一个段落,搜索 P2P 网络中与查询最匹配的 top *k* 个文档.

本文提出了一种基于直方图的分层 top-*k* 查询算法.首先,我们采用层次化的方法实现分布式的 top-*k* 查询.利用向量空间模型(vector space model,简称 VSM)[4]进行本地的 top-*k* 查询,然后聚集搜索到的所有节点的 top-*k* 结果产生最优的 top-*k* 结果.由于 P2P 中每个查询过程都可以用一棵查询树来表示,我们可以分层进行 top-*k* 查询.儿子节点返回 top-*k* 结果给父亲节点,父亲节点聚集所有儿子节点的 top-*k* 结果和本地的 top-*k* 结果产生最优的 top-*k* 结果.分层的 top-*k* 查询将结果的排序和合并分散到网络中的各个节点,充分利用了网络中的计算资源.由于结果自底向上进行合并,只取合并后的 top-*k* 个结果向上传递,一些不相关或者较差的结果就在合并的过程中被过滤掉,因此节省了网络带宽.其次,我们根据 peer(节点)返回的结果为 peer 构建直方图,利用直方图为将来的查询估计 peer 可能的分数上限,对 peer 进行选择,裁剪查询树的一些无用分枝(不包括真正的 top-*k* 结果),提高了搜索的效率.父亲节点为每个儿子节点建立直方图,直方图也是分层的.由于我们只根据儿子节点返回的结果建立直方图,因此不需要儿子节点事先建立直方图.随着查询的进行,直方图可以自动更新,是自适应的.维护直方图的代价也很小.本文对基于直方图的分层 top-*k* 算法进行了性能分析,并用实验证明该算法是有效的.

1 分层的 Top-*k* 查询

由于 P2P 系统是一个分散的、动态的系统,系统中不存在中心控制节点,节点可以动态地加入和退出网络,因此,P2P 环境下的 top-*k* 查询必须以一种完全分散的方式进行,同时还要考虑节点的动态加入和退出.我们首先给出 P2P 环境下的 top-*k* 查询模型,然后基于模型提出分层的 top-*k* 查询算法.

1.1 纯的P2P环境下的Top-*k*查询模型

纯的 P2P 网络(如 Gnutella)的搜索是一种广播式的搜索,节点除了进行本地搜索还将查询被广播发送给它的所有邻居节点.搜索范围由查询跳数 *TTL* 给出,查询每转发一次,*TTL* 减 1,当 *TTL*=0 时,就停止传播查询.为了防止回路的产生,每个查询消息都有一个唯一的编号.如果节点先前已经收到查询,则说明发生了回路,节点不再继续传播查询.我们可以把这一查询过程用一棵查询树来表示,如图 1 所示.图中节点 a 发起查询,为查询树的根节点.查询被广播发送给节点的所有邻居节点,每次转发 *TTL* 减 1,当 *TTL*=0 或者没有邻居的节点为叶子节点.

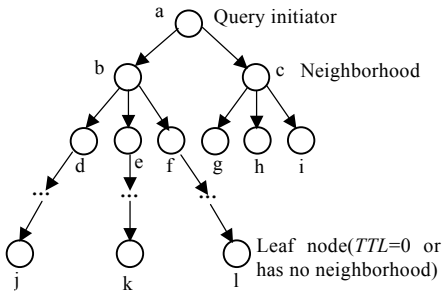


Fig.1 Query processing tree

图 1 查询树

下面,我们给出查询树的形式化定义.

定义 1. 假设节点 *P* 为查询 *Q* 的发起节点,*TTL* 为查询跳数,则查询树 *QueryTree*(*P*,*Q*,*TTL*)可以递归定义为:

(1) 节点 P 为查询树 $QueryTree(P,Q,TTL)$的根节点;

(2) 假如 TTL=0 或者节点 P 没有邻居节点(不包含节点 P 的父亲节点),则查询树 $QueryTree(P,Q,TTL)$只包含一个节点 P;

(3) 否则,假设节点 $P_1,P_2,\ldots,P_n$ 为节点 P 的邻居节点(不包含节点 P 的父亲节点),查询 Q 被广播发送到节点 $P_1,P_2,\ldots,P_n$.$QueryTree(P_1,Q,TTL-1)$,$QueryTree(P_2,Q,TTL-1)$,$\ldots$,$QueryTree(P_n,Q,TTL-1)$为查询树 $QueryTree(P,Q,TTL)$的子树,它们的根节点分别为 $P_1,P_2,\ldots,P_n$.节点 P 为 $P_1,P_2,\ldots,P_n$ 的父亲节点.为了防止回路的产生,假如节点 $P_i(1\le i\le n)$先前已经收到过查询 Q,则删除节点 P 到节点 P_i 的回路.

定义了查询树,我们定义 P2P 下的 top-k 查询为搜索查询树中与查询最匹配的 top k 个结果.由于我们将查询的范围集中在非结构化文档上,top-k 查询就是搜索与查询最匹配的 top k 个文档.将查询树中所有节点的全部文档作为一个文档集,top-k 查询可以看成是搜索文档集中与查询最为相似的 top k 个文档.

我们利用 VSM 进行 top-k 查询,top-k 函数采用文档和查询的相似度函数.根据相似度函数给文档打分,取分数最高的前 k 个文档作为 top-k 结果返回.文档和查询的相似度通过计算文档向量和查询向量的夹角余弦进行衡量,具体公式为

$$Sim(d_j,q)=\frac{\vec{d}_j\cdot\vec{q}}{|\vec{d}_j|\times|\vec{q}|}=\sum_{i=1}^{n}w_{i,j}\times w_{i,q} \tag{1}$$

其中,$\vec{d}_j$,$\vec{q}$ 分别为文档 d_j 和查询 q 的向量表示,$\vec{d}_j=(w_{1,j},w_{2,j},...,w_{n,j})$,$\vec{q}=(w_{1,q},w_{2,q},...,w_{n,q})$,$w_{i,j}$ 和 $w_{i,q}$ 分别对应词 t_i 在文档 d_j 和查询 q 中的权重.向量之间的夹角余弦通过计算向量的点积得到.

文档中词的权重一般采用 TF*IDF 规则产生,TF(term frequency)表示词频,IDF(inverse document frequency)表示逆文档频率.具体计算公式为

$$w_{i,j}=\frac{f_{i,j}\times\log(N/n_i)}{\sqrt{\sum_{t_i\in d_j}\left[f_{i,j}\times\log(N/n_i)\right]^2}} \tag{2}$$

其中,$w_{i,j}$ 为词 t_i 在文档 d_j 中的权重,$f_{i,j}$ 为词 t_i 在文档 d_j 中的词频,N 为文档的总数,n_i 为文档集中出现 t_i 的文档数,分母为归一化因子.

查询向量的计算相对简单,查询中关键字的权重要么是 1 要么是 0,如果该关键字在查询中出现,则权重为 1,否则为 0.同样,我们也对查询向量进行归一化处理.

式(2)中,计算文档中词的权重需要一些文档集信息.比如,文档的总数和文档集中出现词 t_i 的文档数.在集中式索引中,这些信息都很容易获得.但在 P2P 环境下,由于网络是分散的、动态的,不存在中心控制节点,节点可以随时加入和退出网络,要统计所有节点的文档集信息是十分困难的,节点也不一定会提供自己的文档集信息.因此,更为现实的做法是各个节点根据自己的本地信息进行 top-k 查询,然后聚集所有节点的 top-k 结果产生最优的 top-k 结果.我们将网络中的每个节点看成是一个搜索引擎,节点利用 VSM 进行本地的 top-k 查询.假设不同节点返回的文档分数可以直接比较,定义 P2P 环境下的 top-k 函数为

$$Top\text{-}k(P,Q,TTL)=\max_k(\{Local\text{-}Top\text{-}k(P_i,Q)|P_i\in QueryTree(P,Q,TTL)\}) \tag{3}$$

其中,P 为查询 Q 的发起节点,TTL 为查询跳数,P_i 为查询树 $QueryTree(P,Q,TTL)$所覆盖的节点,$Local\text{-}Top\text{-}k$ 为节点 P_i 的本地 top-k 函数,采用 VSM 的余弦相似度函数(式(1))作为本地 top-k 函数,返回本地与查询最为匹配的 top k 个文档.$\max_k$ 表示聚集查询树中所有节点的 top-k 结果,取其中分数最大的 k 个文档作为最终的 top-k 结果返回.

1.2 分层的Top-k查询算法

求解 top-k 函数(式(3)),我们采用分而治之的方法.由于查询过程可以用一棵查询树来表示,我们可以基于这棵查询树分层计算 top-k 函数,将结果的排序和合并分布到网络中的各个节点上,实现分布式的 top-k 查询.具体递归计算公式如下:

$$Top\text{-}k_1(P,Q,TTL)=\max_k(Local\text{-}Top\text{-}k(P,Q),Top\text{-}k_1(P_1,Q,TTL-1),\ldots,Top\text{-}k_1(P_n,Q,TTL-1)) \tag{4}$$

$$Top\text{-}k_1(P,Q,0)=Local\text{-}Top\text{-}k(P,Q) \tag{5}$$

其中,$P_1,P_2,\ldots,P_n$ 为节点 P 的邻居节点,在查询树 *QueryTree*(*P*,*Q*,*TTL*)中表现为节点 P 的儿子节点.将查询广播发送给节点 $P_1,P_2,\ldots,P_n$ 分别进行 top-*k* 查询.节点 P 除了进行本地的 top-*k* 查询,还要聚集儿子节点 $P_1,P_2,\ldots,P_n$ 返回的 top-*k* 结果,产生最优的 top-*k* 结果.叶子节点(*TTL* 为 0 或者没有邻居的节点)仅仅进行本地的 top-*k* 查询,将 top-*k* 结果返回给它的父亲节点.整个查询过程自底向上逐层进行,直到根节点得到最终的 top-*k* 结果.例如:图 1 中,节点 *j* 查询本地信息返回本地的 top-*k* 结果给父亲节点.节点 b 聚集节点 d,e,f 返回的 top-*k* 结果以及本地的 top-*k* 结果,产生最优的 top-*k* 结果,继续返回给节点 a.节点 a 聚集节点 b 和节点 c 返回的 top-*k* 结果以及本地的 top-*k* 结果,产生最终的 top-*k* 结果,返回给用户.

定理 1. 分层的 top-*k* 查询是正确的,即式(4),(5)等价于式(3).

利用数学归纳法可以很容易地证明.证明从略.

下面,我们给出基于广播搜索的分层 top-*k* 查询算法.

算法 1. 基于广播搜索的分层 top-*k* 查询算法 broadcastTopKQuery.

输入:top-*k* 查询 *query*,跳数 *TTL*,*k* 为返回结果的个数,*P* 为执行查询的节点.

输出:查询树中最优的 top-*k* 结果.

```
P.broadcastTopKQuery(query,TTL,k)
begin
  localResult=localTopKQuery(query,k);
  if (TTL==0) then return(localResult);
  for each of my neighbors P_i do
  begin
    if P_i is not visited then
      peerResult_i=P_i.broadcastTopKQuery(query,TTL−1,k);
  end
  overallResult=MergeResult(localResult,peerResult_1,peerResult_2,…,peerResult_n,k);
  return(overallResult);
end
```

2 利用直方图进行 Peer 选择

算法 1 采用广播的方式遍历查询树,显然这是十分低效的,我们希望在查询传播的过程中能够事先判断出那些不包含真正 top-*k* 结果的节点,从而缩小搜索的范围.我们根据 peer 返回的 top-*k* 结果为 peer 构建直方图,利用直方图为将来的查询估计 peer 可能的分数上限,即查询树中以 peer 为根的子树所包含的文档的分数上限,对 peer 进行选择,裁剪查询树中的一些无用分枝(不含真正的 top-*k* 结果),以提高搜索的效率.

2.1 直方图的构建

在介绍如何构建直方图之前,我们首先介绍相似度函数的一个性质.

性质 1. 假设文档 d_j 和查询 q 的向量都经过归一化处理,则相似度函数 $Sim(d_j,q)=\sum_{i=1}^{n} w_{i,j}\times w_{i,q}$ 是单调递增的,即给定两个文档 d_1 和 d_2, $\vec{d}_1=(w_{1,1},w_{2,1},\ldots,w_{n,1})$, $\vec{d}_2=(w_{1,2},w_{2,2},\ldots,w_{n,2})$.假如对所有的 $w_{i,q}\neq 0$,都有 $w_{i,1}\geq w_{i,2}$,则 $Sim(d_1,q)\geq Sim(d_2,q)$.

假设 peer 返回的 top-*k* 结果不仅包含文档的分数,而且包含文档的向量表示.根据性质 1,我们可以利用 peer 返回的 top-*k* 查询结果构建 peer 的直方图.由于 P2P 环境下的查询与查询跳数相关(见第 1.1 节),因此为 peer 构建不同跳数的直方图.设查询 q 在 peer A 上的跳数为 *TTL*,peer A 返回的与查询 q 最相似的 top-*k* 文档为 $\{d_1,d_2,\ldots,d_k\}$. $\vec{d}_j=(w_{1,j},w_{2,j},\ldots,w_{n,j})$ 为文档 $d_j(1\leq j\leq k)$的向量表示, $\vec{q}=(w_{1,q},w_{2,q},\ldots,w_{n,q})$ 为查询 q 的向量表示. $\vec{d}_j$, $\vec{q}$ 都已归一化.对查询 q 中出现的关键字 $t_i, w_{i,q}\neq 0$,我们计算 t_i 在 top-*k* 文档中的权重上限,即 $w_{i,A,TTL}=\max_{1\leq j\leq k} w_{i,j}$,并构建 peer A 跳数为 *TTL* 的直方图为 $Histogram_{A,TTL}=\{(t_i,w_{i,A,TTL}=\max_{1\leq j\leq k} w_{i,j})\mid w_{i,q}\neq 0\}$.

计算 peer A 相对于查询 q 的分数上限 $UpperScore_{q,A,TTL}=\sum_{w_{i,q}\neq 0} w_{i,A,TTL}\times w_{i,q}$.由性质 1,可以保证 $UpperScore_{q,A,TTL}\geq Sim(d_j,q),1\leq j\leq k$.由于 $d_j(1\leq j\leq k)$是 peer A 返回的与查询 q 最相似的前 k 个文档,代表了以 peer A 为根的子树 $QueryTree(A,q,TTL)$的最优 top-k 结果,因此 $UpperScore_{q,A,TTL}$ 一定大于等于以 peer A 为根的子树 $QueryTree(A,q,TTL)$所包含的所有文档的分数上限.假设在 TTL 范围内没有节点的加入和退出,节点中也没有相关文档的加入和删除,则相对于查询 q 而言,直方图是正确的.

注意,这里的直方图是一种特殊类型的直方图,不同于统计学的直方图.它为查询中的每个关键字存储其在文档中的权重上限,用于为将来的查询估计 peer 可能的分数上限,从而进行 peer 选择.

构建了直方图,我们可以利用直方图对今后的查询 q'估计 peer A 的分数上限,假设查询 q'在 peer A 上的跳数为 TTL', $\vec{q}'=(w_{1,q'},w_{2,q'},...,w_{n,q'})$ 为查询 q'的向量表示,计算 peer A 相对于查询 q'的分数上限为

$$UpperScore_{q',A,TTL'}=\sum_{t_i\in Histogram_{A,TTL}\wedge TTL=\min\{TTL_1|TTL_1\geq TTL'\}} w_{i,A,TTL}\times w_{i,q'}+\sum_{\neg\exists Histogram_{A,TTL}(t_i\in Histogram_{A,TTL}\wedge TTL\geq TTL')} 1\times w_{i,q'} \quad (6)$$

假如 peer A 的直方图中存在跳数大于等于 TTL'的直方图并且包含关键字 t_i,我们取跳数与 TTL'最接近的直方图中的权重上限 $w_{i,A,TTL}$ 计算 peer A 的分数上限;否则,我们取最大权重 1 计算 peer A 的分数上限.由于经过归一化的词的权重取值范围在 0~1 之间,如式(2)所示,取最大权重 1 能够保证估计的分数上限一定大于实际的分数上限.

利用直方图估计 peer 相对于查询 q'的分数上限是不精确的.一方面,当查询 q'中的关键字都不在直方图中出现时, $UpperScore_{q',A,TTL}=\sum_{i=1}^{n}1\times w_{i,q'}$,显然大于实际 peer 的文档分数上限;另一方面,根据查询 q 的 top-k 结果建立的直方图对查询 q 是正确的,但对查询 q'却不一定正确,直方图中关键字 t_i 的权重上限 $w_{i,A}$ 可能小于实际查询结果中关键字 t_i 的权重上限.大多数情况下,直方图估计的 peer 分数上限均大于实际的分数上限.

与 top-k 查询相似,直方图也是分层的.父亲节点为每个儿子节点建立直方图,估计每个儿子节点的分数上限.由于我们只根据儿子节点返回的结果建立直方图,因此不需要儿子节点事先建立直方图.随着查询的进行,直方图可以自动更新,是自适应的.

2.2 直方图的更新

由于我们根据查询返回的 top-k 结果构建直方图,直方图可以随着查询的进行自动更新,维护直方图的代价很小.更新直方图的过程和创建直方图的过程相似,根据 peer A 返回的 top-k 查询结果,计算每个查询关键字 t_i 在 top-k 文档中的权重上限 $w'_{i,A,TTL}$.假如 peer A 跳数为 TTL 的直方图 $Histogram_{A,TTL}$ 中不包含的关键字 t_i,则加入$(t_i,w'_{i,A,TTL})$到直方图 $Histogram_{A,TTL}$ 中;否则,比较直方图中关键字 t_i 的权重上限 $w_{i,A,TTL}$ 和 $w'_{i,A,TTL}$.假如 $w'_{i,A,TTL}>w_{i,A,TTL}$,则修改直方图中关键字 t_i 的权重上限为 $w'_{i,A,TTL}$,否则保持 $w_{i,A,TTL}$ 不变.

由于直方图中关键字的权重上限是历史上限,随着查询的执行,权重上限值将会越来越大.由式(6)可知,peer 分数上限的估计值 $UpperScore_{q',A,TTL}$ 也会越来越大.由于我们根据 peer 的分数上限进行 peer 选择,直方图的这种累计效应会影响到查询的效率,导致访问过多的无效节点;另一方面,由于 P2P 系统是一个动态的系统,节点上的内容可以自由地改变.一些原来不含 top-k 结果的节点可能现在包含了 top-k 结果.因此,我们需要定期刷新直方图,保证直方图信息的准确性.刷新直方图由一个后台进程进行.刷新的方法类似于查询,对 peer A 跳数为 TTL 的直方图 $Histogram_{A,TTL}$ 中包含的所有关键字 t_i,发送仅含关键字 t_i 的查询到 peer A,查询跳数为 TTL,进行 top-k 查询.采用基于广播搜索的 top-k 查询算法(算法 1),再根据 peer A 返回的 top-k 结果更新直方图.由性质 1,关键字 t_i 在 $Histogram_{A,TTL}$ 中的新的权重上限 $w'_{i,A,TTL}$ 一定是以 peer A 为根的子树 $QueryTree(A,t_i,TTL)$所包含的文档中 t_i 的权重上限.直方图刷新的频率可以根据节点当时的负载决定,选择节点负载低的时候进行刷新.

2.3 基于直方图的Peer选择

利用直方图估计了 peer 可能的分数上限,我们可以根据 peer 的分数上限对 peer 进行选择.首先对 peer 进行排序,依次选取分数最好的 peer 发送查询.假如当前返回的 top-k 文档的最小分数大于等于余下 peer 的分数上限,说明余下的 peer 不包含真正的 top-k 结果,停止继续发送查询,得到最优的 top-k 结果.

修改算法 1,我们得到基于直方图的分层 top-k 查询算法 selectedTopKQuery.

算法 2. 基于直方图的分层 top-k 查询算法 selectedTopKQuery.

输入:top-*k* 查询 *query*,跳数 *TTL*,*k* 为返回结果的个数,*P* 为执行查询的节点.

输出:查询树中最优的 top-*k* 结果.

P.selectedTopKQuery(*query*,*TTL*,*k*)

begin

localResult=localTopKQuery(*query*,*k*);

if (*TTL*==0) **then return**(*localResult*);

for each of my neighbors P_i **do**

$peerUpperScore_i$=estimatePeerScore(P_i,*query*, $TTL-1$);

根据 $peerUpperScore_i$ 从大到小的次序对 P_i 进行排序,假设排好序的 peer 序列为{P_1,P_2,…,P_n};

overallResult=*localResult*;

for (*i*=1;*i*<=*n*;*i*++) **do**

begin

minScore=minTopKScore(*overallResult*);

if (*minScore*>=$peerUpperScore_i$) **and** (Count(*overallResult*)>=*k*) **then return**(*overallResult*);

if P_i is not visited **then**

begin

$peerResult_i$=P_i.selectedTopKQuery(*query*,*TTL*−1,*k*);

updateHistogram($peerResult_i$,*query*,*TTL*−1);

overallResult=MergeResult(*overallResult*,$peerResult_i$,*k*);

end

end

return(*overallResult*);

end

可以看出,算法 2 避免了将查询广播发送到所有的邻居节点,而是根据 peer 可能的分数上限对 peer 进行选择,依次发送查询.当节点的分数上限小于等于当前 top-*k* 文档的最小分数时,以该节点为根的子树不包含真正的 top-*k* 结果,这些节点就被过滤掉,从而提高了搜索的效率.由于直方图根据 peer 返回的结果进行构建,因此开始执行时每个节点都保证被访问到,相当于是广播地发送查询.随着查询的执行,直方图将会根据节点返回的 top-*k* 结果建立和更新,利用直方图估计 peer 可能的分数上限,筛掉一些无用的节点,提高了搜索的效率.

由第 2.1 节的讨论可知,利用直方图估计 peer 的分数上限不是精确的.它可能过大,导致访问一些无用的节点(不包含真正的 top-*k* 结果),也可能过小,导致漏掉一些节点,不能找到所有的 top-*k* 结果.大多数情况下,直方图估计的 peer 分数上限均大于实际的分数上限.实验证明,直方图的方法是有效的,它能在基本不影响查询效果的情况下,大大提高查询的效率.

2.4 节点的加入和退出

P2P 网络的一个重要特性就是网络是动态的,节点可以自由加入和退出网络.在 P2P 的环境下进行 top-*k* 查询必须考虑节点的动态加入和退出.

根据算法 2,我们可以很容易地处理节点的加入和退出.对新加入网络的节点 *A*,由于不存在该节点的直方图,由式(6),节点 *A* 相对于查询 *q* 的分数上限为 $UpperScore_{q,A}=\sum_{i=1}^{n}1\times w_{i,q}$, $UpperScore_{q,A}\geq 1$.由于文档的分数在 0 和 1 之间,根据算法 2,一定保证访问节点 *A*,再根据节点 *A* 返回的 top-*k* 结果,更新节点 *A* 的直方图.对离开网络的节点 *A*,根据算法 2,继续搜索其他节点,直到满足查询终止条件为止.例如:图 1 中,假如节点 e 退出网络,节点 d 发现节点 e 没有响应,将把查询发送给节点 f(前提是当前 top-*k* 结果的最小文档分数小于节点 f 的分数上限),继续搜索.由于我们为每个儿子节点(邻居节点)建立直方图,儿子节点退出,父亲节点可以继续保留儿子节点的直方图信息或者隔一段时间后将儿子节点的直方图删除,更新操作十分简单.我们将在第 4 节实验中测试节点的

动态加入和退出对查询效果的影响.

3 性能分析

本节我们给出基于直方图的分层 top-k 查询算法(算法 2)的性能分析.我们主要考虑如下指标:存储代价、网络通信代价、查询响应时间和更新代价.

3.1 存储代价

我们主要考虑每个节点上存储直方图的代价.由第 2.1 节的讨论,我们知道每个节点根据儿子节点(邻居节点)返回的 top-k 结果,为儿子节点构建直方图.假设节点共有 $N_{neighbor}$ 个邻居节点,每个邻居节点有 N_{TTL} 个不同跳数的直方图,N_{term} 为直方图中的关键字个数,关键字的平均长度为 Len_{term},则节点所需要的存储空间为

$$Storage_{histogram}=N_{TTL}\times N_{neighbor}\times N_{term}\times(Len_{term}+8) \tag{7}$$

其中,直方图中每个关键字对应一个关键字权重的上限,假设为 double 类型,占 8 个字节.由于直方图只需存储查询关键字,因此 N_{term} 较之文档中包含的索引词数目很小.随着查询数目的增多,N_{term} 可能逐渐变大,我们可以规定 N_{term} 的最大数目,并采用最近最少使用 LRU(least recently used)策略进行淘汰,保证直方图中存储常用的关键字.对不出现在直方图中的关键字,我们采用最大权重 1 计算 peer 的分数上限,见式(6).由于我们为 peer 创建不同跳数的直方图,因此存储代价还依赖于跳数的多少.我们可以限制最大的跳数来进一步限制直方图的存储量.

3.2 网络通信代价

网络的通信代价用需要传输的字节数来衡量.不考虑文档下载的传输量,它依赖于文档的大小.我们只考虑查询过程(发出查询到找到结果)所需的传输量.假设查询 K 个文档,查询访问的节点数为 N_{peer},查询包含 $N_{queryterm}$ 个关键字,查询关键字的平均长度为 $Len_{queryterm}$,则总的传输字节数为

$$Communication=(N_{peer}-1)\times N_{queryterm}\times Len_{queryterm}+(N_{peer}-1)\times K\times(8+(Len_{queryterm}+8)\times N_{queryterm}) \tag{8}$$

这里,我们只考虑了实际需要传输的内容,不考虑打包后的消息头、IP 头等附加信息.消息传播过程中可能产生的回路也被忽略.等式右边的前半部分$(N_{peer}-1)\times N_{queryterm}\times Len_{queryterm}$ 为查询传输的总字节数,后半部分为 top-k 结果传输的总字节数.返回的结果不仅包含文档的分数,假设为 double 类型,占 8 个字节,而且包含查询关键字在文档中的权重,假设权重为 double 类型,占 8 个字节.

显然,网络的通信代价与查询访问的节点数成正比,减少访问的节点数能够节约网络的带宽.

3.3 查询相应时间

查询响应时间是用户最为关心的指标之一.在 P2P 网络中,查询响应时间包括网络延迟和访问节点的查询处理时间两个部分.算法 2 中,我们采用串行的方式进行 top-k 查询,收到上一个节点返回的 top-k 结果后才继续将查询发送给下一个节点.查询响应时间为

$$Time_{query}=N_{peer}\times Time_{processing}+ConnectionTime\times(N_{peer}-1)+\frac{Communication}{Bandwidth} \tag{9}$$

$$Time_{processing}=Time_{peerselect}+Time_{localtopk}+Time_{resultmerge}+Time_{updatehistogram} \tag{10}$$

其中,式(9)右边的前半部分 $N_{peer}\times Time_{processing}$ 为访问节点的查询处理时间,后半部分为网络延迟.网络延迟分为两个部分,一部分是建立连接的时间,另一部分是传输数据的时间.$ConnectionTime$ 为建立一个连接的时间,N_{peer} 表示访问的节点数,$Communication$ 为总共传输的数据量,计算见式(8),$Bandwidth$ 为网络的平均带宽.$Time_{processing}$ 是单个节点处理查询的时间,P2P 网络中不同节点 CPU 的处理能力是不一样的,我们取平均处理时间进行估计.$Time_{processing}$ 进一步又可以细分为 peer 选择的时间 $Time_{peerselect}$,本地查询的时间 $Time_{localtopk}$,结果合并的时间 $Time_{resultmerge}$ 和维护直方图的时间 $Time_{updatehistogram}$.下面,我们给出 $Time_{peerselect}$,$Time_{localtopk}$,$Time_{resultmerge}$ 和 $Time_{updatehistogram}$ 的时间复杂度分析.

Peer 选择首先需要根据直方图估计 peer 可能的分数上限.对查询中的每个关键字,我们取跳数与当前查询跳数最为接近的直方图中的权重上限估计 peer 的分数上限,见式(6).假设 $N_{neighbor}$ 为邻居节点的个数,$N_{queryterm}$ 为查询关键字的个数,N_{term} 为直方图中关键字的个数,每个节点具有 N_{TTL} 个不同跳数的直方图,则估计 peer 分数

上限的时间复杂度为 $O(N_{neighbor}N_{queryterm}(N_{TTL}+N_{term}))$.估计了 peer 的分数上限,还要根据 peer 的分数上限对 peer 排序,时间复杂度为 $O(N_{neighbor}\log N_{neighbor})$.因此,$Time_{peerselect}$ 的时间复杂度为 $O(N_{neighbor}N_{queryterm}(N_{TTL}+N_{term})+N_{neighbor}\log N_{neighbor})$.

本地查询的时间取决于与查询相关的本地文档的数目 N_{doc}.假设通过倒排索引检索相关文档,检索时间小于排序时间,$Time_{localtopk}$ 的时间复杂度为 $O(N_{doc}\log N_{doc})$.

结果合并将儿子节点返回的 top-k 结果合并排序,假设 $N_{selectedneighbor}$ 为选择发送查询的邻居节点数,K 为查询返回的文档数,$Time_{resultmerge}$ 的时间复杂度为 $O(N_{selectedneighbor}K\log K)$.

根据查询返回的 top-k 结果更新直方图.首先计算查询关键字在 top-k 文档中的权重上限,然后查找跳数为 TTL 的直方图中是否存在该关键字,更新直方图中的权重上限(如第 2.2 节所述).假设 K 为查询返回的文档数,$N_{queryterm}$ 为查询包含的关键字数目,N_{term} 为直方图中的关键字数目,$Time_{updatehistogram}$ 的时间复杂度为 $O(N_{queryterm}K+N_{queryterm}N_{term})$.

由上面的分析可以看出,串行执行的情况下查询响应时间和访问的节点数成正比.我们可以适当地增加 top-k 查询执行的并发度来缩短查询的响应时间,但并行地执行查询可能会导致访问更多的无效节点,影响搜索的效率.我们将在第 4 节实验中测试提高查询并发度对搜索效率的影响.

3.4 更新代价

由于直方图根据节点返回的 top-k 结果进行更新,因此更新的代价很小,计算公式为

$$UpdateCost = N_{peer}\times HistogramUpdate_{peer} \tag{11}$$

其中,N_{peer} 为访问的节点数,$HistogramUpdate_{peer}$ 为单个节点更新直方图的代价,由第 2.2 节的讨论可知,$HistogramUpdate_{peer}$ 的时间复杂度为 $O(N_{queryterm}K+N_{queryterm}N_{term})$.其中,$K$ 为查询返回的文档数,$N_{queryterm}$ 为查询包含的关键字数,N_{term} 为直方图中包含的关键字数.

从上面的分析我们可以看出,减少访问的节点数可以提高搜索的效率,减少网络的带宽以及更新的代价.串行的执行会影响查询的响应时间,我们可以通过适当增加并发度提高查询的响应时间.基于直方图的分层 top-k 查询算法通过直方图估计 peer 的分数上限,对 peer 进行选择,避免了将查询广播发送到所有邻居节点,减少了访问的节点数,因而提高了查询的效率.由于直方图根据节点返回的 top-k 结果进行,因此直方图是自适应的,更新代价很小.直方图带来的额外存储空间也很小,我们可以通过限制最大跳数以及运用 LRU 策略淘汰直方图中不常用的关键字,进一步减少直方图所需的存储空间.实验证明,直方图的方法是有效的,它能在基本不影响查询效果的前提下,较大地提高查询的效率.实验结果将在第 4 节给出.

4 实 验

本节通过模拟实验验证基于直方图的分层 top-k 查询算法的有效性.我们将对算法的查询效果和查询效率进行实验分析.

4.1 实验设置

所有实验都是在一台 PC 机上完成的.PC 机的配置为 CPU P4 1.6GHz,内存 1GB,操作系统 Windows XP.模拟程序由 JAVA 编写.网络拓扑结构基于 power-law,由 PLOD[5]算法产生.研究显示,Gnutella 的网络拓扑结构接近 power-law[6],因此我们的模拟尽量接近现实中的 P2P 网络.以下是实验涉及的一些参数.

注意:PLOD 算法产生的网络是一个无向图,每个节点平均出度为 3.6,相当于无向图中有 1 800 条无向边.

Top-k 查询的文档集采用 SMART[7]系统中所使用的测试文档集 MED,CACM 和 Cranfield.其中,MED 包括 1 033 个医疗类文档摘要和 30 个查询,CACM 包含 3 204 个计算机类文档摘要和 64 个查询,Cranfield 包含 1 398 个空气动力学类文档摘要和 225 个查询.由于篇幅有限,我们只给出 MED 的测试结果,其余文档集的测试结果与 MED 类似.文档集的分布采用 even(均匀)和 80-20 两种分布.Even 分布将文档集均匀地分布在网络中的各个节点上;而 80-20 分布将 80%的文档分布在 20%的节点上,余下的 20%的文档分布在 80%的节点上.两种分布都将

文档集划分给网络中的各个节点,文档不允许重复存放在多个节点上.

Table 1 Parameter and settings

表 1 参数设置

Parameter name	Default value	Description
Network topology	power-law	The topology of network, with outdegree of 3.6
Network size	1 000	The number of peers in the network
TTL	5	The time-of-live of a query message
k	10	The number of documents returned

4.2 查询效果

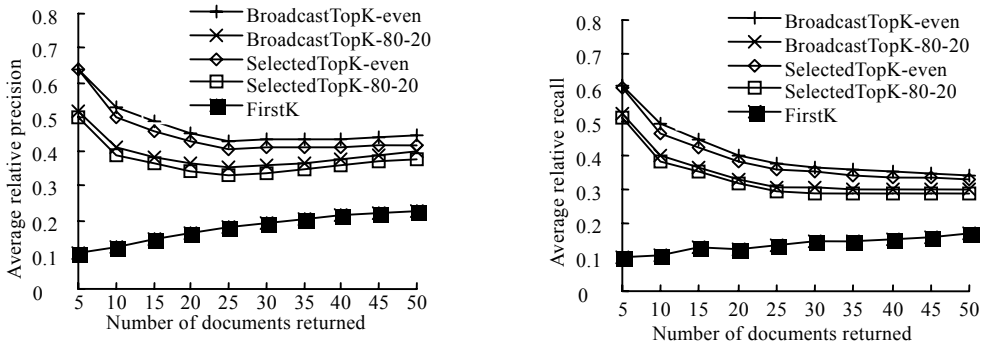
判断 top-*k* 查询的查询效果,我们采用信息检索中的查准率(precision)和查全率(recall)来加以衡量.定义集中式索引(采用 VSM 模型)的 precision 和 recall 作为理想的最好结果,计算相对 precision 和 recall 作为 top-*k* 查询效果的衡量标准.计算公式如下:

$$\text{相对 precision}=\text{precision}/\text{集中式索引的 precision} \tag{12}$$

$$\text{相对 recall}=\text{recall}/\text{集中式索引的 recall} \tag{13}$$

4.2.1 实验 1

我们测试基于直方图的分层 top-*k* 查询的查询效果.执行 MED 的所有 30 个查询,每个查询执行 10 次,每次查询从网络中任选一个节点进行,最后计算相对平均 precision 和 recall.图 2 给出了 1 000 个节点的 P2P 网络在返回不同文档数的情况下的相对平均 precision 和 recall.



(a) Precision (a) 查准率 (b) Recall (b) 查全率

Fig.2 Relative average precision and recall of top-*k* query against number of documents returned

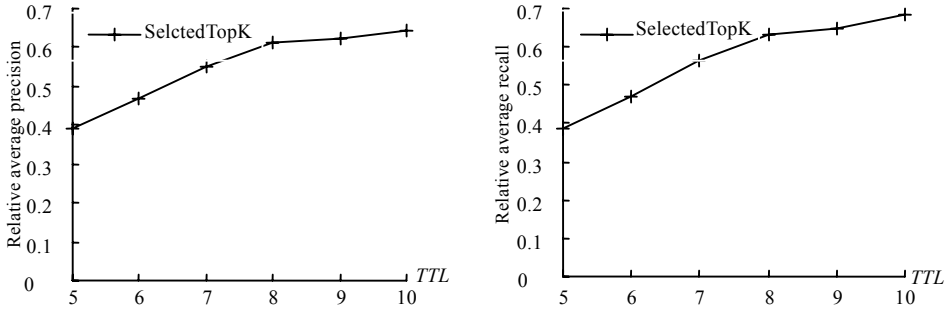
图 2 Top-*k* 查询在返回不同文档数的情况下的相对平均查准率和查全率

图 2 中,BroadcastTopK 表示基于广播搜索的分层 top-*k* 查询算法(算法 1),SelectedTopK 表示基于直方图的分层 top-*k* 查询算法(算法 2),FirstK 表示取到 *k* 个与查询相关的结果就停止搜索.首先,我们比较了 top-*k* 和 FirstK 的查询效果,可以看出,top-*k* 的查询效果明显好于 FirstK.实验证明,top-*k* 查询提高了用户的满意度,是有效的.其次,我们比较了算法 1 和算法 2 的查询效果,可以看出,二者的查询效果十分接近,说明基于直方图的方法是有效的,基本不会影响 top-*k* 查询的查询效果.最后,我们还比较了 top-*k* 查询在均匀分布和 80-20 分布情况下的查询效果,均匀分布下的查询效果略好于 80-20 分布,但相差不大.这是由于 top-*k* 查询搜索跳数在 *TTL* 范围内的所有节点的最优 top-*k* 结果,因此,不同的文档分布不会对查询造成太大的影响.由于 80-20 分布更加符合现实 P2P 系统中的情况,下面的实验我们只考虑 80-20 分布,不再考虑均匀分布.

4.2.2 实验 2

由于 top-*k* 查询的搜索范围受到 *TTL* 的限制,不能访问 *TTL* 之外的节点.增大 *TTL*,扩大 top-*k* 查询的搜索范围,可以提高 top-*k* 查询的查准率和查全率,实验结果如图 3 所示.*TTL* 取 10 时,广播式搜索平均可以访问到 700 个节点(一方面由于网络中存在很多回路;另一方面由于 power-law 的网络中少量的节点拥有很大的出度,大部分节点只有很小的出度),查准率是集中式索引的 64.2%,查全率是集中式索引的 68.5%.增加访问节点的数目,查

询效果还将增加.纯 P2P 环境下的 top-*k* 查询只是搜索一个局部范围内(查询树覆盖的节点)的最优,并不是全局的最优.



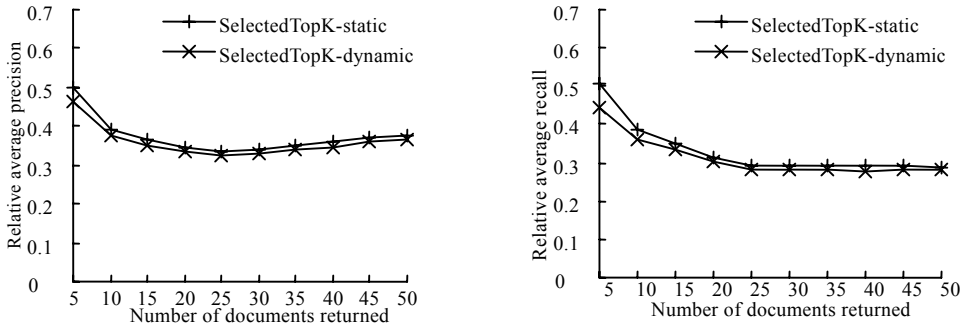
(a) Precision (a) 查准率 (b) Recall (b) 查全率

Fig.3 Relative average precision and recall of top-*k* query against different *TTL* (*K*=10)

图 3 Top-*k* 查询在不同 *TTL* 情况下的相对平均查准率和查全率(*K*=10)

4.2.3 实验 3

由于 P2P 网络是一个动态的网络,节点可以自由地加入和退出网络.下面,我们测试在节点动态加入和退出情况下的 top-*k* 查询效果.同样,执行 MED 的 30 个查询,每个查询执行 10 次,每次查询从网络中任选一个节点进行,查询的过程中随机选择一定数量的节点(不多于 10 个)退出或加入网络.节点加入网络时,任意选择 3 个节点作为自己的邻居节点.图 4 给出了动态环境下的基于直方图的分层 top-*k* 查询(算法 2)的相对平均查准率和查全率.在保证网络中节点数目相对稳定(加入和退出的节点数基本相等)的情况下,动态网络的 top-*k* 查询效果和静态网络的 top-*k* 查询效果相当,说明算法 2 能够较好地适应网络动态改变的情况.



(a) Precision (a) 查准率 (b) Recall (b) 查全率

Fig.4 Relative average precision and recall of top-*k* query in dynamic network environment

图 4 Top-*k* 查询在动态网络环境下的相对平均查准率和查全率

4.3 查询效率

由第 3 节的讨论得知,减少访问的节点数可以提高搜索的效率,减少网络的带宽以及更新的代价.因此,我们取查询访问的节点数 N_{peer} 作为衡量查询效率的指标.

4.3.1 实验 4

我们测试基于直方图的分层 top-*k* 查询的效率.执行 MED 的所有 30 个查询,每个查询执行 10 次,每次查询从网络中任选一个节点进行,最后计算平均访问的节点数,结果如图 5 所示.我们比较了基于广播搜索的分层 top-*k* 查询算法(算法 1)和基于直方图的分层 top-*k* 查询算法(算法 2)所访问的节点数.在执行算法 2 之前,我们首先为每个查询建立直方图.可以看出,基于直方图的 peer 选择有效地裁剪了搜索空间,减少了节点的访问数目,大大提高了查询的效率.由于我们采用的直方图是权重上限直方图(如 2.1 节所述),随着返回结果数的增多,直方图的效果有所减弱,需要访问更多的节点,但仍然比广播方式的搜索效率要高.

4.3.2 实验 5

算法 2 采用串行的执行,影响了查询的响应时间.下面,我们测试提高查询的并发度,将查询同时发送给两个分数最高的儿子节点,并行处理 top-*k* 查询.由于采用的是模拟的实验,因此不测试查询的响应时间.假设网络中每个节点的处理能力相同,可以估算提高并发度后的 top-*k* 查询在最好的情况下能够缩短一半的查询响应时间,最坏的情况也和串行执行的效果一样,平均可以缩短四分之一的响应时间.不过,并行的执行也影响了查询的效率,访问了更多的无效节点.图 6 给出了并发度为 2 的 top-*k* 查询的查询效率.可以看出,提高并发度在一定程度上降低了搜索的效率(访问了更多的节点).需要选择一个合适的并发度,在缩短查询响应时间的同时,兼顾查询的效率.

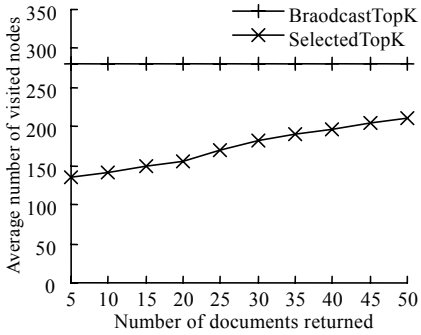


Fig.5 Average number of visited nodes of top-*k* query

图 5 Top-*k* 查询平均访问的节点数

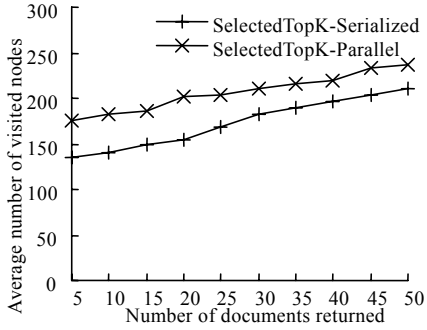


Fig.6 Influence of efficiency under parallel query processing

图 6 并发处理对查询效率的影响

5 相关工作

文献[8]讨论了如何在分布式非协作的环境下进行 top-*k* 查询,提出了用直方图进行数据库的选择.文中给出了 4 种直方图:LA(Linear approximation), MHIST-2(MAXDIFF MHIST-2),FQ(frequent queries)和 ILA (independent linear approximation).其中 LA,MHIST-2 和 ILA 直方图的建立需要本地系统的协作,而 FQ 直方图根据常用查询的结果建立,不需要本地系统的协作.本文的直方图(以下简称为 Upper-bound 直方图)与 FQ 直方图类似,也是根据查询结果建立的,是自适应的.但 Upper-bound 直方图分布在 P2P 网络中的各个节点上,父亲节点为儿子节点建立直方图,直方图是分层的.而 FQ 直方图集中维护在中心控制节点上,只有 1 层.Upper-bound 直方图和 FQ 直方图的构建方式也不一样,Upper-bound 直方图为查询中的每个关键字存储在其文档中的权重上限,而 FQ 直方图只根据常用的查询构建直方图,直方图中存储与查询最为匹配的前 *k* 个元组.Upper-bound 直方图比 FQ 直方图粒度更细,对直方图中没有出现过的查询,Upper-bound 直方图仍然可以进行估计.

分布式信息检索中采用 GLOSS[9]和 CORI[10]方法对文档集进行选择,选取其中包含相关文档最多的前 *k* 个文档集发送查询,它们不能保证检索得到的 top-*k* 结果一定是最优的 top-*k* 结果.本文采用分层的方法进行 top-*k* 查询.假设各个节点返回的结果可以直接比较,可以保证搜索得到的 top-*k* 结果一定是查询树中最优的 top-*k* 结果.利用 peer 返回的 top-*k* 结果构建直方图,估计 peer 可能的分数上限,即以 peer 为根的子树包含的文档的分数上限,提高了查询效率.实验证明,直方图的方法是有效的,它能在基本不影响查询效果的基础上,提高了查询的效率.

文献[11]采用层次化的方法实现多媒体数据库上的 K-NN 查询,类似于算法 1,将查询广播发送给所有的邻居节点,搜索效率低下.文献[12]利用路由索引提高搜索的效率,但维护索引的代价很高,不适于经常动态改变的 P2P 网络.文献[13]提出了几种在 P2P 网络中提高搜索效率的策略,核心思想是尽量减少接收和处理查询的节点.然而,top-*k* 查询要求搜索尽可能多的节点得到与查询最相似的结果.

现有一些 P2P 环境下的信息检索系统:PlanetP[14]基于向量空间模型(VSM)进行 P2P 环境下的信息检索,但它需要节点事先利用 bloom filter 对节点上的内容进行摘要,然后将摘要广播发送到网络中的各个节点,可扩展

性较差.pSearch[15]结合了分布式 HASH 表(DHT),VSM 和潜在语义索引(latent semantic indexing)进行 P2P 下的信息检索.它基于 CAN[16]系统,将文档的向量表示看作是坐标空间中的一个点,利用 DHT 进行节点定位,但它需要一些全局的统计信息,比如整个网络的 IDF(逆文档频率)信息.PeerIS[17]通过 peer 聚类提高了搜索的效率,但它需要每个 peer 提供各自的元数据信息,通过一个集中控制节点将 peer 分配到相应的类中.这些系统都没有考虑如何在纯的 P2P 环境下进行 top-*k* 查询.

6 结 论

本文讨论了如何在 P2P 环境下进行有效的 top-*k* 查询,检索与查询最为相关的 top-*k* 个文档.提出了一种基于直方图的分层 top-*k* 算法.实验证明,该算法是有效的,top-*k* 查询提高了查询的精度,而直方图则在基本不影响查询效果的情况下提高了搜索的效率.本文还给出了算法的性能分析,通过分析,直方图的维护代价很小,是有效的.文献[18]提出了一种邻居节点自配置的方法,根据节点返回的结果对节点的邻居进行动态配置,将那些返回结果最多的节点调整成为查询节点的直接邻居,从而更快地找到结果.研究如何利用返回的 top-*k* 结果调整邻居节点,进一步提高 top-*k* 查询的效果和效率是我们下一步研究的目标.

References:

[1] Napster Home Page. http://www.napster.com/

[2] Gnutella Home Page. http://www.gnutella.com/

[3] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review, 2001,31(4):149−160.

[4] Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Boston: Addison Wesley, 1999. 27−30.

[5] Palmer CR, Steffan JG. Generating network topologies that obey power law. In: Proc. of the GLOBECOM. San Francisco: IEEE, 2000. 434−438. http://citeseer.ist.psu.edu/palmer00generating.html

[6] Ripeanu M. Peer-to-Peer architecture case study: Gnutella network. Technical Report, TR-2001-26, University of Chicago, 2001.

[7] Buckley C. Implementation of the SMART information retrieval system. Technical Report, TR35-686, Cornell University, 1985.

[8] Yu C, Philip G, Meng WY. Distributed top-*n* query processing with possibly uncooperative local systems. 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. 117−128.

[9] Gravano L, Garcia-Molina H, Tomasic A. The effectiveness of GLOSS for the text database discovery problem. In: Snodgrass RT, Winslett M, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1994. 126−137.

[10] Callan JP, Lu ZH, Croft WB. Searching distributed collections with inference networks. In: Proc. of the 18th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Seattle: ACM Press, 1995. 21−28.

[11] Zhang W, Li JZ, Gao H, Pan HW. K-NN query processing in peer-to-peer multimedia database. Computer Science, 2002,29(8):190−193.

[12] Crespo A, Garcia-Molina H. Routing indices for peer-to-peer systems. In: Proc. of the 22nd Int'l Conf. on Distributed Computing Systems. Vienna: IEEE Computer Society, 2002. 23−34. http://dbpubs.stanford.edu:8090/pub/2001-48

[13] Yang B, Garcia-Molina H. Efficient search in peer-to-peer networks. In: Proc. of the 22nd Int'l Conf. on Distributed Computing Systems. Vienna: IEEE Computer Society, 2002. 5−14. http://dbpubs.stanford.edu:8090/pub/2001-47

[14] Cuenca-Acuna FM, Nguyen TD. Text-Based content search and retrieval in ad hoc P2P communities. Technical Report, DCS-TR-483, Department of Computer Science, Rutgers University, 2002.

[15] Tang CQ, Yu ZH, Mahalingam M. pSearch: Information retrieval in structured overlays. ACM SIGCOMM Computer Communication Review, 2003,33(1):89−94.

[16] Ratnasamy S, Francis P, Handley M, Karp RM, Schenker S. A scalable content-addressable network. ACM SIGCOMM Computer Communication Review, 2001,31(4):161−172.

[17] Ling B, Lu ZG, Ng W, Ooi BC, Tan KL, Zhou AY. A content-based resource location mechanism in PeerIS. In: Proc. of the 3rd Int'l Conf. on Web Information Systems Engineering. Singapore: IEEE, 2002. 279−289. http://citeseer.ist.psu.edu/ling02contentbased.html

[18] Ng W, Ooi BC, Tan KL, Wang XY, Ling B, Zhou AY. A novel peer-to-peer system based on self-configuration. Journal of Software, 2003,14(2):237−246 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/237.htm

附中文参考文献:

[11] 张炜,李建中,高宏,潘海为.基于 Peer-to-Peer 的多媒体数据库 K-NN 查询处理.计算机科学,2002,29(8):190−193.

[18] 黄维雄,黄铭钧,陈建利,王晓宇,凌波,周傲英.一种基于自配置策略的新型 peer-to-peer 平台系统.软件学报,2003,14(2):237−246. http://www.jos.org.cn/1000-9825/14/237.htm

****

2005 年全国开放式分布与并行计算学术会议

征 文 通 知

由中国计算机学会开放系统专业委员会主办、上海大学计算机学院承办、上海计算机学会协办的“2005 年全国开放式分布与并行计算学术会议”将于 2005 年 10 月 27 日－29 日在上海召开，有关信息如下：

一、征文范围(包括但不限于)

开放式分布与并行计算模型及体系结构；

下一代开放式网络、数据通信、网络与信息安全、业务管理技术；

开放式海量数据存储与 Internet 索引技术，分布与并行数据库及数据/Web 挖掘技术；

开放式机群计算、网格计算、Web 服务、P2P 网络及中间件技术；

开放式移动计算、自组网与移动代理技术；

分布式人工智能、多代理与决策支持技术；

分布与并行计算算法及其在科学与工程中的应用；

开放式虚拟现实技术与分布式仿真；

开放式多媒体技术，包括媒体压缩、内容分送、缓存代理、服务发现与管理技术。

二、征文要求

详见会议主页：http://www.cs.shu.edu.cn/DPCS2005 可查询进一步的会议信息。

三、重要日期

会议时间：2005 年 10 月 27～29 日

截稿日期：2005 年 7 月 15 日

录用通知：2005 年 7 月 30 日

四、联系方式

投稿地址：200072 上海延长路 149 号上海大学计算机学院 缪淮扣 收(请在信封上注明 DPCS2005)

电　　话：021-56331669

电子邮件: bfzhang@staff.shu.edu.cn (请在邮件主题中注明 DPCS2005)