渐进/分布式网页聚类算法 PG+与 PG++*

王启新, 李 毅, 董 丽, 聂 宇, 王克宏

(清华大学 计算机科学与技术系,北京 100084)
E-mail: liyi1997@mails.tsinghua.edu.cn
http://www.tsinghua.edu.cn

摘要: 用户行为分析是 Web 站点信息推荐中的重要方法,被广泛应用在该领域的诸多算法中.PageGather 算法是其中有代表性的一种.旨在解决静态 PageGather 算法输入数据量过大、时间复杂度高的问题,使其更具实用性.通过引入渐进学习和分布的机制,给出了改进的算法 PG+和 PG++,并进行了实验分析.改进后,既保证了算法的等效性,又明显提高了效率.

关 键 词: Web;渐进;分布式;PageGather;聚类
中图法分类号: TP393 文献标识码: A

当前 Web 站点信息推荐的方法大致可分为如下几个类型:(1) 基于兴趣模型的半自动化类型.典型的系统有 WebWatcher[1],AVANTI[2]等.过多的人工参与是这类系统的缺点.(2) 自动用户行为分析类型.又分为用户浏览路径分析和网页相关性分析两种子类型.前者的典型方案有 Footprints[3]、阳小华的方案[4]等.后者的典型算法有 PageGather 算法(简记 PG 算法)[5]等.(3) 合作过滤类型.典型的系统有 Ringo[6],GroupLens[7]等.这种系统也需要过多的人工参与.(4) 与结构化信息相结合的类型.典型的系统有 STRUDEL[8]等.该类系统可移植性差,需要对现有的网站结构作大规模的调整,甚至破坏.(5) 客户端个性化类型.典型的系统有 Letizia[9],PINS[10]等,这类系统可与服务器端的个性化系统相结合,构建更高层次的二元的个性化系统.

综合考虑自动化程度、技术可行性和结果质量,目前自动用户行为分析类型的方案最具有竞争力.该类方案中有代表性的算法之一便是 PG 算法[5].文献[11]讨论了其诸多的优点.

但是,PG 算法是静态的,训练集一次性给出,在大规模网站的实际应用中往往会遇到用时过长的问题.我们的思路是采取渐进学习和分布式计算的机制来解决数据量大与运算时间有限的矛盾,从而提出了 PG+和 PG++算法.

1 静态算法描述及其实现

定义 **1.1**. 一个浏览过程是指用户在一次上网过程中为了某一目的对一个网站的网页进行访问的序列.

文献[12]中也称浏览过程为“事务(transaction)”,称上网过程为“会话(session)”.一个会话是由若干个事务首尾连接而成的.

定义 **1.2**. 对同一网站的任意两个网页 P_i,P_j,在一次浏览过程中,如果已知访问了 P_i,则记该浏览过程也访问了 P_j 的概率为 $P(P_j|P_i)$(这里与文献[5]的描述略有不同,其原因请参见文献[13]);同理,定义 $P(P_i|P_j)$,称 $\min\{P(P_i|P_j),P(P_j|P_i)\}$为 P_i,P_j 的合同访问概率,并将其作为 P_i,P_j 的相关度.

* 收稿日期: 2001-06-04; 修改日期: 2001-09-07

作者简介: 王启新(1977－),男,上海人,硕士,主要研究领域为知识工程,数据挖掘,知识发现;李毅(1978－),男,北京人,硕士生,主要研究领域为知识工程,Web 挖掘,知识发现;董丽(1974－),女,河南郑州人,讲师,主要研究领域为知识工程;聂宇(1978－),男,北京人,助理研究员,主要研究领域为知识工程;王克宏(1941－),男,江苏镇江人,教授,博士生导师,主要研究领域为知识工程,Web 挖掘,知识发现.

© 中国科学院软件研究所 http://www.jos.org.cn

定义 **1.3**. 相关度矩阵(similarity matrix)$A\{i,j\}$的元素 a_{ij} 表示第 i 号和第 j 号网页间的相关度.

显然,相关度矩阵是一个对称方阵,且对角元素全为 1.

PG 算法.

(1) 将网站 Log 记录划分成独立的浏览过程.文献[14]中对浏览过程的划分作了详细的分析并提出了一些建议.一种较为简单的划分方式是:认为每一个会话对应一个浏览过程,且每一个 IP 地址对应一个用户.首先找出同一个用户的所有 Log 记录,并按时间排序,得到序列$\{P_i\}$,其中每一个元素是访问的一个网页.我们观察序列$\{P_i\}$中两两前后相邻的元素的访问时间间隔,凡是小于设定时间长度阈值的(例如 1 个小时),就认为同属于一个会话.

(2) 计算两两网页间的合同访问概率,并以此作为该两网页间的相关度,构建相关度矩阵.

记 $U_i=\{v|v$ 是含有网页 P_i 的浏览过程$\}$,$U_j=\{v|v$ 是含有网页 P_j 的浏览过程$\}$,实际操作时,我们用以下公式估计 $P(P_i|P_j)$:

$$\hat{P}(P_i \mid P_j)=\frac{\left\| U_i \cap U_j \right\|}{\left\| U_j \right\|}. \tag{1}$$

我们用同样的方法估计 $P(P_j|P_i)$.我们把$\|U_i\|$,$\|U_j\|$分别称为网页 P_i,P_j的相关浏览过程数,把$\|U_i \cap U_j\|$称为 P_i,P_j的合同访问次数.文献[13]中提出了"最小相关浏览过程数约束"的要求,并给出了其数学上合理性的证明,所以,我们用下式估计合同访问概率:

$$q=\max\{\|U_i\|,\|U_j\|\},$$
$$f_{ij}=\begin{cases}0, & 当 q<T_r,\\ \min\{\hat{P}(P_i \mid P_j),\hat{P}(P_j \mid P_i)\}=\dfrac{\|U_i \cap U_j\|}{q}, & 当 q\ge T_r,\end{cases} \tag{2}$$

其中 T_r 是指定的最小相关浏览过程数阈值.我们把 f_{ij} 称为 P_i,P_j的合同访问频率,作为对 P_i,P_j 相关度的估计.

(3) 在相关度矩阵所定义的图(矩阵元素 a_{ij} 表示图中 i,j 顶点间的关系紧密程度)中进行距离聚类.将相关度高的网页聚在一起.我们采用团聚类方式.虽然寻找最大团的问题是 NP-完全问题,但是求其较优解的算法却可以控制在 k 的多项式时间内完成,k 是人为限定的团的最大元素个数.

(4) 为每一个找到的聚类构建一个相应的 Index 页面,指向该聚类中的所有网页.

实现 PG 算法的数据结构如下:

定义 *TWebPages* 为一链表,用来组织存储 PG 算法作为学习集的网站的网页.这些网页可以是最近 K_p 天内新生成的网页.*TWebPages* 的每一个元素(下标从 0 开始)为一个网页的相关信息,并认为该网页的 ID 就是它在 *TWebPages* 中对应元素的下标.记 *TWebPages* 的元素个数为 N.

将 PG 所使用的过去 K_l 天的日志看做是一个链表,每一个元素是一条 Log,称这个链表为 Log.我们另外定义一个链表 *TAccess*,它是链表 Log 删除了不关心的网页的 Log 之后所得到的子链表.所谓"不关心的网页"是指没有在 *TWebPages* 中出现的网页.记 *TAccess* 的元素个数为 L.

在不引起混淆的情况下,我们用"链表名[i]"来表示访问链表的第 i 个元素.

用矩阵 $M_{N\times N}$ 记录合同访问次数,其元素 m_{ij} 是 ID 为 i 和 j 的网页的合同访问次数.由于 $M^T=M$,且 $m_{ii}(0\le i<N)$ 没有意义,故仅存储 $m_{ij}(0\le i<j<N)$.存储时,按照行主存储的方式将 M 的上半三角(不含对角线)存储在字节流中.用长度为 N 的数组 C_{PRV} 记录相关浏览过程数——$C_{PRV}[i]$ $(0\le i<N)$是 ID 为 i 的网页的相关浏览过程数.

根据 M 和 C_{PRV} 的定义,结合式(2),可得相关度矩阵 A 的元素 $f_{ij}(0\le i<j<N)$的计算公式为

$$q=\max\{\|U_i\|,\|U_j\|\}=\max\{C_{PRV}[i],C_{PRV}[j]\},$$
$$f_{ij}=\begin{cases}0, & 当 q<T_r,\\ \dfrac{\|U_i \cap U_j\|}{q}=\dfrac{m_{ij}}{q}, & 当 q\ge T_r.\end{cases} \tag{3}$$

故实践中,A 可用哈希表存储.

后文用 M 和 A 表示概念上的合同访问次数计数矩阵和相关度矩阵,在不引起歧义时,也有可能指它们真正被存储的部分,请结合上下文理解.

在上述变量及数据结构中,*TWebPages* 和 Log 作为每次算法运行的输入,由用户提供;其他变量及数据结构(N,*TAccess*,M,C_{PRV},A)均在算法运行时动态生成,算法运行结束时自动释放.

© 中国科学院软件研究所 http://www.jos.org.cn

2 渐进式算法——PG+

2.1 算法描述

在静态算法中,每次运行都需要重新生成 M 和 C_{PRV},其每个元素的初值都为 0.不难看出,其中有大量的工作是重复的.因此,我们想到应在永久介质中存储 M 和 C_{PRV},以保留过去的工作成果.我们将 $M(C_{PRV})$改为一个固定规模(N 为常量)的方阵(数组),即总是关心距当前最近的 N 个网页.这 0~N–1 共 N 个网页的 ID 将被循环利用,每天新生成的网页(设有 n 个)将替换 $M(C_{PRV})$中最古老的 n 个 ID,将这 n 个 ID 在 $M(C_{PRV})$所对应的原历史记录清零.做完这样的初始化后,以前 Log 中的浏览过程的合同访问次数和相关浏览过程计数已经存留在 M 和 C_{PRV} 的初值中,我们只要将当天 Log 中的浏览过程计数到 M 和 C_{PRV} 中去就可以了.这样就从输入过去 K_l 天的 Log 减少到只需输入当天的 Log,实现了渐进化.我们称该渐进算法为 PG+算法.

下面我们来重新设计数据结构和算法伪码.

定义 N 作为 PG+算法所处理的网页个数(即每次运行 PG+算法时,只取时间上距离当前最近的 N 个网页进行处理).定义 *TWebPages* 为一个循环链表,它的元素个数恒为 N 个(下标从 0~N–1),每一个元素代表一个网页,并认为该网页的 ID 就是该元素的下标.记 I_b 是算法开始运行前所处理的 N 个网页中最古老的网页的 ID.设 *NewPages* 为当天新加入的网页的数组.我们从第 I_b 号网页开始替换(用新的网页替换老的网页)*TWebPages* 的元素,直到所有新的网页都进入到 *TWebPages* 循环链表为止.记循环链表中最后一个被替换的网页标号之后的下一个标号为 I_e(亦即下一次运行算法时的 I_b).设第 1 次运行渐进算法前 I_b=I_e=0.不妨将最新一天的日志看做是一个链表,每一个元素是一条 Log.称这个链表为 Log.另外定义一个链表 *TAccess*,它是链表 Log 删除了不关心的网页的 Log 后所得到的子链表.这里所谓"不关心的网页"是指用 *NewPages* 的元素更新过后,没有在 *TWebPages* 中出现的网页.仍记 *TAccess* 的元素个数(即 PG$^+$算法所处理的 Log 条数)为 L.在不引起混淆的情况下,仍用"链表名(或循环链表名)[i]"来表示访问链表的第 i 个元素.与第 1 节相同,仍用矩阵 $M_{N\times N}$ 记录合同访问次数,存储方法亦不变,仍用长度为 N 的数组 C_{PRV} 记录相关浏览过程数——C_{PRV} [i](0≤i<N)是 ID 为 i 的网页的相关浏览过程数.相关度矩阵 A 的生成仍参照式(3).

对于上述变量及数据结构,N 是常量,其值永久存储在程序代码中;*NewPages* 和 Log 作为每一次算法运行的输入,由网站管理者提供;*TAccess* 链表和 I_b 在算法运行时动态生成,算法运行结束时自动释放;其他变量及数据结构(*TwebPages*,I_e,M,C_{PRV})均存储在永久介质(磁盘)上,每一次算法运行开始时将首先调入上一次算法运行所得出的结果对它们进行赋值.

PG+算法.

```
1.  //步骤 0:初始化
2.  读入存储在永久介质中的变量及数据结构:TWebPages,Ie,M,CPRV;
3.  //步骤 1:网页信息入库——更新 TwebPages 循环链表以及 Ib,Ie
4.  Ib=Ie;
5.  int i=Ib; j=0;
6.  while (j<NewPages.length){
7.        将 TWebPages 中 ID 为 i 的网页的相关记录更新为 NewPages[j]所代表的网页.
8.        j++;
9.        i=(i+1)%N;
10.   }
11.  Ie=i;
12.  //步骤 2:Log 入库及浏览过程辨识
13.  建链表 TAccess=∅;
14.  for (int i=0; i<Log.length; i++){
15.       int j=Log[i]所记录的访问的网页在 TWebPages 中的编号.
16.           if ( j 不存在) continue;
17.           else 将 Log[i]添加到 TAccess 末尾;
18.      }
19.      分析 TAccess 中的浏览过程,标记其每一个元素(log 记录)属于哪一个浏览过程;
```

© 中国科学院软件研究所 http://www.jos.org.cn

20. //步骤 3:M 相关行列、C_{PRV} 相关元素清零
21. 将 M 的第 I_b(含) ~ I_e(不含)行和列清零(注意,从 I_b 到 I_e 是循环链表的下标顺序遍历关系,即每次加 1 后还要模 N,参见第 9 行)
22. 将 C_{PRV} 的第 I_b(含) ~ I_e(不含)元素清零;
23. //步骤 4:合同访问及相关浏览过程数计数
24. 根据 *TAccess* 中所记录的浏览过程,增加 M 中相应合同访问次数和 C_{PRV} 中相应相关浏览过程数的计数.方法同 PG 算法步骤 3.
25. //步骤 5:生成相关度矩阵 A
26. 根据式(3)生成相关度矩阵 A;
27. //步骤 6:聚类
28. 根据 A 中所记录的相关度进行网页聚类.

2.2 等效定理

第 1 节中已经提到了一个网页是"被关心的"说法,为了下文描述方便,我们给出更规范的定义:

定义 **2.1**. 称一个网页是被关心的,如果该网页是 PG(或 PG+,PG++)算法所处理的网页,它与其他网页的相关度被计算,最后参与聚类算法.否则称这个网页是不被关心的.

定义 **2.2**. 称一条 Log 是被关心的,如果这条 Log 所记录访问的网页是被关心的网页,否则称该条 Log 是不被关心的.

运行完 PG+算法步骤 1(参见第 2.1 节)之后,M 和 C_{PRV} 中所关心的发布时间最古老的和最新的网页也就被确定了.用 t_b 表示该最古老的网页的发布时间,用 t_e 表示最新的网页的发布时间.在 PG+算法历次运行累积输入的 Log 中,记最古老的 Log 的生成时间为 t_b',记最新的 Log 的生成时间为 t_e'.假定 PG+算法历次运行所累积的网页和 Log 输入中不遗漏任何新生成的网页和 Log.

定理 **2.1**. 如果 PG+算法 P'在某次运行中完成了步骤 1 后确定关心且仅关心时间段$[t_b,t_e]$内新生成的网页,并且完成了步骤 2 后确定历次(含本次)运行所累积输入的 Log 为时间段$[t_b',t_e']$内的 Log,则其运行结果与关心且仅关心$[t_b,t_e]$内新生成的网页,并以时间段$[t_b',t_e']$内的 Log 作为学习集的 PG 算法 P 的结果是相同的.

关于定理的说明性论证,请参见文献[13].

由于 t_b 时刻以前的 Log 不会被关心,而 t_e'时刻以后的 Log 又没有被输入,所以被 $P(P')$用作学习集的 Log 是$[\max(t_b,\ t_b'),t_e']$时间段内的 Log(如果 $\max(t_b,t_b')\geq t_e'$,则 Log 学习集为空),我们把$[\max(t_b,t_b'),t_e']$时间段称为 PG(PG+)算法的 Log 学习集时间段.我们不难对定理 2.1 作进一步推广,得到如下定理:

定理 **2.2**(等效定理). 如果 PG+算法在某次运行中完成了步骤 1 后确定关心且仅关心时间段$[t_b,t_e]$内新生成的网页,则其运行结果与关心且仅关心$[t_b,t_e]$内新生成的网页,并使用相同 Log 学习集时间段的 PG 算法的结果是相同的.

3 分布式合同访问计数——PG++

在 PG+算法中,M 和 C_{PRV} 必须存放在永久介质中,以供下一次运行使用.另外,$O(N^2)$的庞大空间规模也决定了 M 必须存放在磁盘这类海量数据存储器上.这就造成了一个问题——磁盘的访问速度远远低于内存,尤其在随机读写上更是如此;而 PG+算法中却存在大量的对 M 的随机读写操作(参见第 2.1 节程序伪码的"步骤 4"),这就成为 PG+算法运行时的一个瓶颈.从第 4 节的表 1 中也可以看出,合同访问计数占用了 3 个多小时,是整个程序运行时间的 54.3%.为了解决这个问题,我们设计了一个分布式的系统,通过提高读写操作的并行度来加速合同访问计数的速度.分布式合同访问计数——PG++.

我们的出发点是通过对矩阵 M 的分割,使得分布系统中不同的节点负责 M 某一部分元素的随机读写,从而分散磁盘随机访问的工作,达到对 PG+算法第 4 步(合同访问次数计数)的并行处理,提高速度.

以下记用于分布式读写的节点个数有 B 个,每个节点从 0~(B–1)进行编号.在不引起歧义的情况下,用 s 表示当前所讨论节点的编号.

© 中国科学院软件研究所 http://www.jos.org.cn

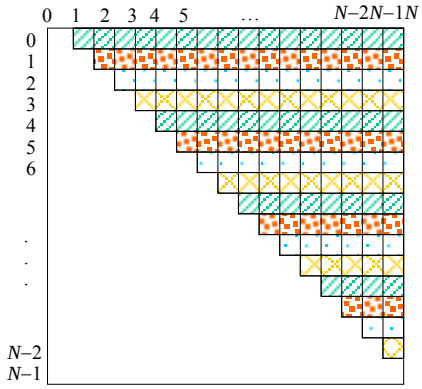

Fig.1
图 1

我们对 M 采取行主基数分割,如图 1 所示(B=4).M 的各行被按照基数 B 分配到 B 个节点上,即编号为 s 的节点负责模 B 为 s 的所有行.

在合同访问计数的过程中,需要根据浏览过程对 M 的元素进行增 1 计数.设浏览过程 v 中不重复的网页编号从小到大排序后为 $v=\{P_0,P_1,\ldots,P_n\}$,则需要对 M 中的所有 $m_{ij}(i,j\in\{P_0,P_1,\ldots,P_n\})$元素进行加 1.这些元素形成 M 的一个子阵,将它称作“增 1 子阵”,记作 $\tilde{M}$.根据 M 的存储方式,实际上只要对 $\tilde{M}$ 的上半三角增 1.在不引起歧义时, $\tilde{M}$ 也可能指其真正被存储的部分(上半三角),具体意义请结合上下文理解.

行主基数分割时,能达到接近于 B^2 的超线性加速比,详细分析请参见第 4.3 节.这样,在渐进式算法 PG+的基础上,我们又实现了分布机制.我们称这个算法为 PG++算法.

4 加速比分析及实验分析

4.1 概述与约定

PG 算法及改进算法中有两步必须用到对外存海量数据的随机访问:(1) Log 入库与浏览过程辨识;(2) 合同访问计数.所以,这两步很自然地成为静态 PG 算法的主要耗时步骤.下面我们就着重分析静态 PG 算法、PG+算法、PG++算法在这两步中的时间复杂度,从而说明为何 PG+算法和 PG++算法较静态 PG 算法有了明显的改进.

为了简化定量分析,假设以下参数均为常量:

(1) 每天新生成的(新闻)网页的数量恒为 n;

(2) 每天新生成的 Log 的条数恒为 l;

(3) 每天新生成的 Log 中所隐含的浏览过程个数恒为 v,其中包含 j 个不同网页的浏览过程的个数恒为 v_j,且 $v=\sum_{j=0}^{\infty}v_j$.为了便于分析,假定每个浏览过程中的 Log 访问的网页互不重复,即包含 j 个不同网页的浏览过程的长度正好为 j.这样,易见有 $l=\sum_{j=0}^{\infty}v_j\cdot j$.

为了便于对比,在运行静态 PG,PG+和 PG++算法时,均关心最近生成的 N 个网页.由于假定每天恒生成 n 个网页,故每次运行 PG+算法或 PG++算法时,均关心且仅关心过去 K 天(含当天, $K=\frac{N}{n}$)内新生成的网页.若 PG+算法或 PG++算法累计运行了 K_l 天,则其 Log 学习集时间段(定义参见第 2.2 节)为过去的 K^*天(含当天).其中

$$K^*=\min(K_l,K). \tag{4}$$

根据等效定理(参见定理 2.2),与之等效的静态 PG 算法应关心过去 K 天新生成的网页,并以过去 K^*天的 Log 作为 Log 学习集.

对于用作训练集的第 0 天、第−1 天、…、第−(K^*−1)天的 Log,我们认为第−i 天的每条 Log 被关心的概率是相同的,记之为 p_i.特别地,根据文献[13]所提出的“新闻网页访问的时间局部性现象”及其数学模型,若处理的网站是新闻网站,则应有:

$$\text{当 } i<K-7 \text{ 时},p_i\approx 1.0,\text{且 } \sum_{i=K-7}^{K-1}p_i\approx 3 . \tag{5}$$

4.2 PG/PG+/PG++算法在Log入库与浏览过程辨识中的加速比分析

辨识 Log 中隐含的浏览过程,根据第 1 节的描述,实际上是将所有的 Log 先按照浏览用户,再按照浏览时刻进行排序,所以即使使用内排序,并使用最好的算法,也需要 $O(L\log L)$的时间复杂度[15],其中 L 是指用作学习集的 Log 的条数.

© 中国科学院软件研究所 http://www.jos.org.cn

用符号 $T_X^{(1)}(L)$ 表示将 L 条 Log 入库并辨识浏览过程的(期望)耗时,其中 X 表示 PG 算法的类型,取 PG,PG+或 PG++算法.由于我们的实验系统在"Log 入库与浏览过程辨识"步骤中使用了商业数据库软件和外存,所以有理由(即便再考虑到 Log 入库的时间复杂度,由于其$\in O(L)$,故也不影响量级)认为实验中 $T_X^{(1)}(L) \in \Omega(L\log L)$ 且 $T_X^{(1)}(L) \in O(L^2)$.

假设 $T_X^{(1)}(L) = \tau L\log L$,则可以计算 PG+算法和 PG++算法在"Log 入库与浏览过程辨识"步骤上与静态 PG 算法相比的加速比 $\alpha_{\mathrm{PG+}}^{(1)}$ 和 $\alpha_{\mathrm{PG++}}^{(1)}$ 如下:

$$\alpha_{\mathrm{PG++}}^{(1)} = \alpha_{\mathrm{PG+}}^{(1)} = (K^*)^2. \tag{6}$$

当取 K^*为其最大值 K 的时候,

$$\alpha_{\mathrm{PG++}}^{(1)} = \alpha_{\mathrm{PG+}}^{(1)} = K^2. \tag{7}$$

在我们的实验系统中,$l \approx 287\ 182, K^* = 4$,见表 1.

Table 1 Runtime comparison among the three methods

表 **1** 3 种算法运行时间对比

	PG	PG+	PG++
Run-time for storing and discriminating the Log①(s)	59 547	9 405	8 969
Minimum of the theoretic estimation of acceleration② $K^*(1+\log_l K^*)$	——	4.44	4.44
Actual acceleration③	——	6.33	6.63
Maximum of the theoretic estimation of acceleration④ K^{*2}	——	16	16

①Log 入库与浏览过程辨识实际耗时,②理论估计最小加速比,③实际加速比,④理论估计最大加速比.

4.3 PG/PG+/PG++算法在合同访问计数中的加速比分析

我们用符号 $T_X^{(2)}$ 表示算法每一次运行需要在 M 中进行合同访问计数的(期望)耗时,其中 X 表示 PG 算法的类型,取 PG,PG+或 PG++这3种算法.

由于文件系统普遍采用内存 Cache 机制,M 又是行主存储,所以可以近似认为随机读写增 1 子阵的一行的耗时可以忽略不计(无论有多少元素),而找到该行则是需要机械定位时间 τ的.对于静态 PG 算法来说,对于第$(-i)$天的每个长度为 j 的浏览过程来说,其中有 jp_i 条 Log 是被关心的(p_i 的定义见式(5)),故对于该浏览过程,其增 1 子阵有 jp_i 行.这样的浏览过程在第$(-i)$天有 v_j 个,而我们用最近 K^*天的 Log 作为训练集,故

$$T_{\mathrm{PG}}^{(2)} \approx \sum_{i=0}^{K^*-1}\sum_{j=0}^{\infty} v_j\, jp_i\tau = \sum_{i=0}^{K^*-1} p_i\Big(\sum_{j=0}^{\infty} v_j\, j\tau\Big).$$

记 $H = \sum_{j=0}^{\infty} v_j\, j\tau$,则上式变为

$$T_{PG}^{(2)} \approx H\sum_{i=0}^{K^*-1} p_i.$$

对于 PG+,每次运行只需对当天的浏览过程进行计数,所以有

$$T_{\mathrm{PG+}}^{(2)} \approx \sum_{j=0}^{\infty} v_j\, jp_0\tau = p_0\Big(\sum_{j=0}^{\infty} v_j\, j\tau\Big) = Hp_0.$$

对于 PG++算法,由于引入了分布机制,每个节点所需要负责的增 1 子阵行数减少到原来的 1/B,另外,由于每个节点所负责的 M 分片的规模都降到了原来(整个 M)的 1/B,换行所需要跨越的元素个数也减少到了原来的 1/B,这意味着磁头机械定位的时间降低到了原来的 1/B,即 τ降低到接近 τ/B.所以有

$$T_{\mathrm{PG++}}^{(2)} \approx \sum_{j=0}^{\infty} v_j \frac{jp_0}{B}\frac{\tau}{B} = \frac{p_0}{B^2}\Big(\sum_{j=0}^{\infty} v_j\, j\tau\Big) = H\frac{p_0}{B^2}.$$

因此我们可以估计 PG+,PG++算法在"合同访问计数"步骤上较之静态 PG 算法的加速比 $\alpha_{\mathrm{PG+}}^{(2)}$ 和 $\alpha_{\mathrm{PG++}}^{(2)}$ 如下:

© 中国科学院软件研究所 http://www.jos.org.cn

$$\alpha_{PG+}^{(2)}=\frac{T_{PG}^{(2)}}{T_{PG+}^{(2)}}=\frac{\sum_{i=0}^{K^*-1}p_i}{p_0}. \tag{8}$$

特别地,当理想情况下,即 $p_i=p_0(0\le i\le K^*-1)$时

$$\alpha_{PG+}^{(2)}=K^*. \tag{9}$$

由于我们的实验系统是新闻网站,故据式(5)、式(8)可得,当 $K^*<K-6$ 时,有

$$\alpha_{PG+}^{(2)}\approx K^*, \tag{10}$$

即 PG+算法将"合同访问计数"的工作量降到了 $1/K^*$,这符合我们的初衷.当 K^*取最大值 K 的时候,根据式(5)、式(8),有

$$\alpha_{PG+}^{(2)}\approx K-3. \tag{11}$$

对于 PG++算法,我们有

$$\alpha_{PG++}^{(2)}=\frac{T_{PG}^{(2)}}{T_{PG++}^{(2)}}=\frac{B^2\sum_{i=0}^{K^*-1}p_i}{p_0}. \tag{12}$$

特别地,当理想情况下,即 $p_i=p_0(0\le i\le K^*-1)$时

$$\alpha_{PG++}^{(2)}=B^2K^*. \tag{13}$$

对于我们的新闻网站实验系统,据式(5)、式(12)可得,当 $K^*<K-6$ 时,有

$$\alpha_{PG++}^{(2)}\approx B^2K^*, \tag{14}$$

即 PG++算法将"合同访问计数"的工作量降到了 $1/B^2K^*$,得到了超线性的加速比.

当 K^*取最大值 K 的时候,根据式(5)、式(12),亦有

$$\alpha_{PG++}^{(2)}\approx B^2(K-3). \tag{15}$$

表 2 是我们的实验数据,它们基本吻合前面对加速比的分析.在我们的实验系统中,取 K=30,分布系统的节点数 B=4,静态 PG 算法采用 K^*=4.

Table 2 Runtime comparison among the three methods

表 2 3 种算法运行时间对比

	Run-Time for counting the relating visitation①	The theoretic estimation of acceleration②	Actual estimation of acceleration③
PG	39 107	——	——
PG+	12 849	$\alpha_{PG+}^{(2)}\approx K^*=4$	3.04
PG++	970	$\alpha_{PG++}^{(2)}\approx B^2K^*=64$	40.03

①合同访问计数实际耗时,②加速比理论估计值,③实际加速比.

References:

[1] Joachims, T., Freitag, D., Mitchell, T. WebWatcher: a tour guide for the World Wide Web. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann Publishers, 1997. 770~775.

[2] Fink, J., Kobsa, A., Nill, A. User oriented adaptivity and adaptability in AVANTI project. In: Proceedings of the Software-Ergonomie'97. Stuttgart: B.G. Teubner, 1997. 135~144.

[3] Wexelblat, A., Maes, P. Footprints: history-rich Web browsing. In: Proceedings of the Conference Computer-Assisted Information Retrieval. 1997. 75~84.

[4] Yang, Xiao-hua, Zhou, Long-xiang. View of Web users. Journal of Software, 1999,7:690~693 (in Chinese).

[5] Perkowitz, M., Etzioni, O. Adaptive Web sites: automatically synthesizing Web pages. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98). Menlo Park, CA: AAAI Press/MIT Press, 1998. 727~732.

[6] Shardanand, U., Maes, P. Social information filtering: algorithms for automating "Word of Mouth". In: Proceedings of the Conference in Human Factors in Computing Systems. Denver, Colorado: ACM Press, 1995. 210~217.

© 中国科学院软件研究所 http://www.jos.org.cn

[7] Resnick, P., Iacovou, N., Suchak, M., *et al*. GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the Conference on Computer Supported Cooperative Work CSCW'94. New York, NY: ACM Press, 1994. 175~186.

[8] Fernandez, M., Florescu, D., Kang, J., *et al*. System demonstration-strudel: a Web-site management system. In: Proceedings of the ACM SIGMOD Conference on Management of Data. 1997. 549~552.

[9] Lieberman, H. Letizia: an agent that assists Web browsing. FS-95-03, Menlo Park, CA: AAAI Press, 1995. 97~102.

[10] Dong, Li, Li, Jing-hua, Wang, Ke-hong. A personalized intelligent navigation system-PINS based on CORBA and Java technologies. In: Proceedings of the 4th International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region. Los Alamitos, CA: IEEE Computer Socity, 2000. 518~521.

[11] Perkowitz, M., Etzioni, O. Towards adaptive Web sites: conceptual framework and case study. Artificial Intelligence, 2000,118(1,2):245~275.

[12] Mobasher, B., Cooley, R., Srivastava, J. Automatic personalization based on Web usage mining. Communications of the ACM, 2000,43(8):142~151.

[13] Wang, Qi-xin. Study and implementation of clustering algorithms for Web site personalization [MS. Thesis]. Beijing: Department of Computer Science and Technology, Tsinghua University, 2001 (in Chinese).

[14] Cooley, R., Mobasher, R., Srivastava, J. Data preparation for mining World Wide Web browsing patterns. Knowledge and Information Systems, 1999,1(1):5~32.

[15] Yan, Wei-min, Wu, Wei-min. Data Structure. 2nd ed, Beijing: Tsinghua University Press, 1992 (in Chinese).

附中文参考文献:

[5] 阳小华,周龙骧.Web 用户的视图.软件学报,1999,7:690~693.

[13] 王启新.Web Site 个性化聚类算法的研究与实现[硕士学位论文].北京:清华大学计算机科学与技术系,2001.

[15] 严蔚敏,吴伟民.数据结构.第 2 版,北京:清华大学出版社,1992.

Incremental and Distributed Web Page Clustering Algorithms PG+ and PG++*

WANG Qi-xin, LI Yi, DONG Li, NIE Yu, WANG Ke-hong

(*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*)

E-mail: leeyi0001@263.net

http://www.tsinghua.edu.cn

Abstract: A user behavior analysis is an important approach in many algorithms for the Web site information recommendation, among which, PageGather is a typical algorithm. However, the original PageGather algorithm is static, which needs too many data inputs and too much computing time. In this paper, incremental learning and distributed computation mechanisms are introduced into PageGather, so that two improved algorithms PG+ and PG++ are proposed. At the same time, corresponding experimental results are presented and analyzed. The improved algorithms are equivalent to the static PageGather algorithms. And better effect has got.

Key words: Web; incremental; distributed; PageGather; clustering

* Received June 4, 2001; accepted September 7, 2001

© 中国科学院软件研究所 http://www.jos.org.cn