基于用户访问树的Web日志挖掘数据预处理

刘加伶[1] 范 军[2]

(重庆工学院 重庆 400050)[1] (重庆邮电大学计算机科学与技术学院 重庆 400065)[2]

摘 要 在Web日志挖掘中数据预处理是整个挖掘过程的基础,直接影响日志挖掘的质量和结果。提出了一种基于用户访问树的Web日志挖掘数据预处理方法,该方法在处理过程中根据Web日志建立用户访问树,并利用用户访问树进行用户和事务识别,从而可以在缺乏网站拓扑结构的情况下准确地对Web日志进行预处理。

关键词 Web日志挖掘,数据预处理,用户识别,事务识别

中图法分类号 TP311 **文献标识码** A

Data Preprocessing in Web Log Mining Based on User Access Tree

LIU Jia-ling[1] FAN Jun[2]

(Chongqing Institute of Technology, Chongqing 400050, China)[1]

(College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)[2]

Abstract Data preprocessing is the basis of the whole process of data mining in Web log mining, which directly influences the quality of the Web log mining and its result. A method of data preprocessing in Web log mining based on the user access tree was proposed. The user access tree was created according to the Web logs in the preprocessing and it was used to identify the user and transaction. So the preprocessing can be worked well without the site topology.

Keywords Web log mining, Data preprocessing, User identification, Transaction identification

1 前言

随着Internet技术的发展,网络资源迅猛增长。如何使Internet用户有效快速地获得所需的资源,已成为网站设计者亟待解决的问题[1]。解决这个问题的途径之一就是Web日志挖掘,即根据Web服务器记录的日志对用户访问网站的情况进行分析,使用数据挖掘技术抽取日志中感兴趣的模式,得到站点的被访问规律[2],从而改进网站的组织结构和服务。Web日志挖掘过程一般分为3个阶段:预处理阶段、模式发现阶段和模式分析阶段。其中数据预处理是Web日志挖掘的首要工作,其过程在数据挖掘过程中占据了约60%的时间[3],由于Web日志挖掘结果的可靠性和准确性在很大程度上取决于来源数据的准确性,因此数据预处理是整个挖掘过程的基础和实施有效挖掘算法的前提。

Web日志挖掘数据预处理的主要目标是从Web服务器日志中准确地识别出访问网站的用户,重建各个用户在本网站的行为序列即用户会话,然后进一步将用户会话分割成有意义的事务,其过程通常包括数据清洗、用户识别、会话识别、路径补充、事务识别几个步骤[4,5]。其中,用户识别、路径补充以及事务识别都需依赖网站拓扑结构才能准确识别[2],通常使用网络爬虫软件crawler获取网站拓扑结构,然而在电子商务(e-commerce)网站、电子学习(e-learning)网站尤其在网页动态生成的网站中,网络爬虫crawler无法使用[6],此时网站拓扑结构是很难获取的。本文在分析国际上Web数据挖掘数据预处理方面研究状况的基础上,针对在Web日志挖掘预处理中缺乏网站拓扑结构的情况,提出了基于用户访问树的Web日志挖掘数据预处理方法,根据服务器中的用户访问日志建立用户访问树,将同一个用户的所有访问请求记录在同一棵访问树中,并在用户访问树的基础上进行用户和事务识别。

2 基于用户访问树的用户数据预处理

Web日志挖掘的数据来源有3种:客户端、代理服务器和服务器端。由于客户端数据不容易搜集,代理服务器端的访问数据只能用来分析通过该代理服务器访问的用户行为,通常进行操作的数据源主要是服务器端的日志。典型的Web站点日志格式遵循W3C标准[7],日志记录中包含以下信息:用户IP地址、时间戳、方法(如GET,POST)、被请求文件的URL、超文本传输协议(HTTP)的版本号、返回码(请求的状态,成功或错误码)、传输字节数、参考页的URL(用户从该页发出当前文件的请求)、代理(用户使用的浏览器和操作系统的类型)等。Web服务器在响应用户的请求时,将用户请求的文件发送出去的同时把本次请求写入日志;所以Web服务器日志详细记录了用户访问本站点的信息。可以将一个站点的所有日志记录表示为集合 $L=\{l_1,l_2,\cdots,l_n\}$,$|L|=n$ 表示日志集合 L 包含的元素数目,其中的每一个记录 $l\in L$ 都为元组 $l_i=\{ip,url,time,refer,\cdots,method,agent\}$。

本文提出一种基于用户访问树的Web日志预处理方法,

到稿日期:2009-01-17 返修日期:2009-04-20

刘加伶(1963—),女,副教授,主要研究方向为信息安全、信息处理与数据库技术等,E-mail:jiall_cq@hotmail.com。

该方法能在没有网站拓扑结构的情况下较好地对 Web 日志进行预处理，主要根据 Web 日志建立用户访问树并用其进行用户和事务识别，从而得到用户事务数据，其基本流程如图 1 所示。

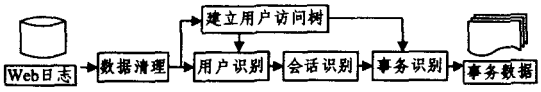


图 1　基于用户访问树的 Web 日志预处理

2.1　数据清理

由于 HTTP 协议是一个无连接协议，用户每下载一个文件，它都会在日志中增加一条记录。通常，用户的一个 HTML 页面请求会产生几条日志记录，因为页面中通常包含对一些图片或其他资源的引用，图片的下载也会在日志中增加一条记录。由于分析处理的主要目的是得到用户的访问模式，因此只保留日志中用户请求的网页地址，图片、音频、脚本文件等日志记录应该删除。基于此，可通过检查 URL 的后缀删除不相关的数据，例如：将 *l. url* 中后缀为 GIF，JPEG，JPG，gif，jpeg，jpg，cgi，JS，js 的请求项日志记录删除，同时，只保留 *l. status* 以 2 和 3 开头以及 *l. method*=*Get* 的记录，因为它们是标识请求成功以及被成功转向请求的记录。

2.2　基于用户访问树的用户识别

用户是指通过一个浏览器访问一个或几个服务器的个体。由于缓存、代理服务器和防火墙的使用，使得准确识别用户的任务变得十分复杂。目前常用的技术是基于日志/站点、拓扑结构的办法进行识别，然而如果网站的拓扑结构比较复杂，再根据拓扑结构识别页与页之间的关系时，效率就会降低，同时在电子商务(e-commerce)网站、电子学习(e-learning)网站尤其在网页动态生成的网站中，网站拓扑结构是很难获取的。为此本文对该策略进行了改进，不利用网站的拓扑结构，而是借助每条日志中 *l. refer* 为 *l. url* 的引用这一关系构建用户访问树，并结合日志记录中的 *l. ip*，*l. agent* 属性，对用户进行识别。

(1)用户访问树的结点存储结构设计

由于从一个页面 URL 可以超链接到多个页面，因此访问树的每个结点可能都有多棵子树，即每个结点都有多个指针域，其中每个指针都指向一棵子树的根结点。同时由于 *l. ip*，*l. agent* 两个属性都相同的记录可能识别出多个用户，为了便于查找，*n* 个头指针组成线性表，采用顺序存储结构。本文设计用户访问树的存储结构说明如下：

```
typedef struct Node{
  unsigned char* pszURL; //记录该结点的 URL
  int nChildrenNum; //记录该结点的孩子数目
  struct Node * pParent; //指向该结点的父结点
  struct Node * pChild[MAX_C_NUM]; //指向以该结点作为参引的结点
}Node, * PNode;
typedef struct{
  int m_nID; //该树的序列号
  PNode m_pRoot; //每棵树的根结点
}CTree;
typedef struct{
  CTree * pTreeSet[MAX_T_NUM]; //头结点指针分别指向每一棵用户拓扑树
  int nTreeNum; //当前拓扑树的数目
}CUserAccessTree;
```

(2)用户访问树的建立过程

用户访问树的建立主要是在 *l. ip* 和 *l. agent* 相同的记录集中进行的，此时根据用户的 IP 和 Agent 信息无法将用户识别出来，然而如果一个请求访问的页面与已经访问过的所有页面之间并没有直接的链接，即该请求页面不在任何一棵用户访问树上，则可以识别为一个新的用户。基于此，根据当前请求 URL 与参考页的 URL 信息建立用户访问树，将同一个用户的所有访问请求置于同一棵树上。首先将所有日志记录按 *l. ip*，*l. agent* 和 *l. time* 属性升序排序，对 *l. ip* 和 *l. agent* 相同的记录集建立用户访问树，建立过程如下：

```
precedure: AccessTreeCreat
input: l. ip 和 l. agent 相同的记录集
output: UserAccessTree
begin
pTreeSet[MAX_T_NUM]初始化为 Null;
for 记录集中的每一条记录 do
    建立结点 NodeUrl, NodeRef;
    If(!(在 pTreeSet 中查找 NodeUrl)) //如果没有找到 NodeUrl 结点
        {
            If(!(在 pTreeSet 中查找 NodeRef)) //如果没有找到 NodeRef 结点
                建立一棵新树，NodeRef 作为根结点;
            NodeUrl 作为 NodeRef 的子结点;
        }
  end_for;
end; //procedure
```

(3)用户识别过程

在建立用户访问树的基础上，用户识别过程如下：

Step1　日志 *l. ip* 不同为不同的用户；

Step2　日志 *l. ip* 相同，*l. agent* 不同则为不同用户；

Step3　日志 *l. ip*，*l. agent* 均相同，建立用户访问树，不同树上的结点即为不同用户访问的记录，所有在一棵树上的记录为同一个用户访问网站被记录的结果。

经过用户识别之后日志的记录 $l_i=\{uid, url, time, refer\}$。其中 *uid* 为识别用户后赋予每个用户的唯一识别码。

2.3　会话识别

当某一个用户的页面请求在时间跨度上较大时，就有可能是用户多次访问了同一个网站，此时可以将用户的访问记录分成多个会话来处理。会话识别的目的就是将同一用户的所有访问序列分成多个单独的用户一次访问序列。用户会话(User Sessions)S 是一个二元组$\langle uid, RS\rangle$，其中 *uid* 是用户标识，RS 是用户在一段时间内请求的 Web 页面的集合。RS 包含用户请求的页面的 URL 和页面的请求时间，则用户会话 S 可以表示为元组：$S=\langle uid, \{(url_1, time_1), \cdots, (url_k, time_k)\}\rangle$[8]。通常采用超时方法识别用户会话，即假定用户一次访问的时间有最大限制，以用户访问历时作为划分会话的分界。即上式中的会话必须满足条件 $time_k-time_1\leqslant T$，T 为超时阈值。根据 perkowitz[9] 统计的结果，一般 T 为

25.5min，业界一般取 30min。

2.4 基于用户访问树的事务识别

在 Web 日志挖掘领域中，用户会话是唯一具备自然事务特征的对象，但它对于关联规则挖掘和序列模式挖掘任务而言粒度仍较大，需要特定的算法将用户会话分割为更小的事务，对用户会话进行语义分组。由于本地缓存和代理服务器的存在以及用户可能使用浏览器的“BACK”按钮，使得服务器的日志会遗漏一些页面请求，因此通常在事务识别之前使用路径补充算法[4]将遗漏的请求页面补充完整。然后采用 Chen[10]等人提出的最大向前引用路径（简称为 MFP）来定义事务。即对于每个用户会话，从开始页面为起点，每个最大前向引用路径都为一个事务。这里将每一个事务都定义为一组页面的访问，从第一次引用开始直到在某处向后回溯为止。

由于用户访问树中已经记录了用户访问的路径信息，因此我们结合会话序列和用户访问树，在不需要补充回退路径步骤的情况下，只要根据用户会话序列遍历用户访问树就可以得到用户的访问事务。

按 *l.uid*，*l.sid*，*l.time* 对以上步骤处理过的日志进行升序排序，对有相同 *l.uid* 和 *l.sid* 的记录集作事务识别处理，*sid* 为会话标识，具体步骤如下：

```
precedure:TransactionIdentification
input:l. uid 和 l. sid 相同的记录集
output:事物 T 的集合
begin
  读取第一条记录，建立结点 NodeUrl;
  Node node=在 l. uid 所在的访问树中查找 NodeUrl;
  Node CurrenNode=node; //node 作为当前结点
  Url 加入到事物 T 中;
  for 记录集中的剩余每一条记录 do
      建立结点 CNodeUrl;
      if(CNodeUrl 是 CurrenNode 的子结点)
          Url 加入到事物 T 中;
      else
            {
              开始一个新的事物，在 l. uid 所在的访问树中查找 CNodeUrl;
              将 node 到 CNodeUrl 的路径中的结点加入新事务;
            }
      CurrenNode=CNodeUrl; // CNodeUrl 为当前结点
    end_for;
end; //procedure
```

对每一个用户会话应用以上算法就可以生成一个或几个用户访问事务，最终按时间排序的所有用户访问事务构成了事务数据集。

基于用户访问树的 Web 日志预处理由于充分利用了 Web 日志的属性信息，根据用户访问日志建立用户访问树，并根据用户访问树进行用户和事务识别，从而可以在缺乏网络拓扑结构的情况下进行 Web 日志挖掘预处理。

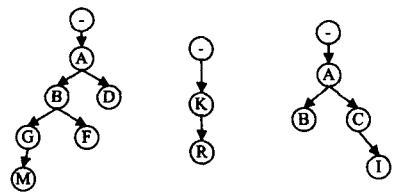
3 实例分析

对表 1 中日志记录集合（已作完数据清理，并按照 *l.ip*，*l.agent* 和 *l.time* 排序，只列出了 *l.ip* 相同的记录集）用基于用户访问树的数据预处理方法进行处理。

表 1 日志记录集合（按照 *l.ip*，*l.agent* 和 *l.time* 排序）

#	IP	Time	URL	Referred	Agent
1	202.192.94.66	19/Oct/2006:11:20:32	A.html	—	IE6.0/WINXP
2	202.192.94.66	19/Oct/2006:11:20:35	B.html	A.html	IE6.0/WINXP
3	202.192.94.66	19/Oct/2006:11:20:39	K.html	—	IE6.0/WINXP
4	202.192.94.66	19/Oct/2006:11:21:32	G.html	B.html	IE6.0/WINXP
5	202.192.94.66	19/Oct/2006:11:23:52	R.html	K.html	IE6.0/WINXP
6	202.192.94.66	19/Oct/2006:11:25:52	M.html	G.html	IE6.0/WINXP
7	202.192.94.66	19/Oct/2006:11:27:31	F.html	B.html	IE6.0/WINXP
8	202.192.94.66	19/Oct/2006:12:20:22	A.html	—	IE6.0/WINXP
9	202.192.94.66	19/Oct/2006:12:20:52	D.html	A.html	IE6.0/WINXP
10	202.192.94.66	19/Oct/2006:11:22:12	A.html	—	IE6.0/WIN98
11	202.192.94.66	19/Oct/2006:11:23:32	B.html	A.html	IE6.0/WIN98
12	202.192.94.66	19/Oct/2006:11:24:11	C.html	A.html	IE6.0/WIN98
13	202.192.94.66	19/Oct/2006:11:26:44	I.html	C.html	IE6.0/WIN98

在用户识别时建立的用户访问树如图 2 所示。



(a)树1 (b)树2 (c)树3

图 2 用户访问树

根据图 2 所示的用户访问树，采用本文的用户识别方法对表 1 中的日志记录识别出 3 个用户的访问路径为 A-B-G-M-F-A-D，K-R 和 A-B-C-I。在会话识别中选定 T 为 30min，可以得到会话识别结果为 A-B-G-M-F，A-D，K-R 和 A-B-C-I。在事务识别过程中，传统的方法首先根据路径补充技术得到会话序列 A-B-G-M-G-B-F，A-D，K-R 和 A-B-A-C-I，然后再利用最大前向引用路径算法得出用户的访问事务为 A-B-G-M，A-B-F，A-D，K-R，A-B，A-C-I。利用第 2.4 节提出的基于用户访问树的事务识别算法，在不需要补充路径的情况便可由用户会话序列直接获得用户的访问事务 A-B-G-M，A-B-F，A-D，K-R，A-B，A-C-I。

从上面的预处理结果可以看出，本文提出的基于用户访问树的 Web 日志挖掘数据预处理方法可以在没有网站拓扑结构的情况下，根据用户访问日志建立用户访问树，并在此基础上对用户进行正确识别；同时因为在数据预处理的过程中根据用户会话序列直接得到用户的访问事务，使得整个 Web 日志预处理只需要做数据清理、用户识别、会话识别、事务识别 4 个阶段的工作。

结束语 随着 Internet 的发展，网络资源更加丰富，Web 日志挖掘已经成为一项重要的研究课题。Web 日志数据预处理是 Web 日志挖掘的一个重要前提和基础，高效正确的预处理方法直接影响着挖掘的成败。本文在对 Web 日志挖掘中的预处理研究的基础上，针对在 Web 日志挖掘预处理中缺乏网站拓扑结构的情况，提出了基于用户访问树的 Web 日志挖掘数据预处理方法，根据服务器中的用户访问日志建立用户访问树，并在用户访问树的基础上进行用户和事务识别。由于 Web 日志挖掘所具有的极大的商业价值和广阔的应用前景以及相关技术还有很大的提升空间，因此将会有更多的

（下转第 210 页）

的回归树进行集成。笔者将这种算法称为 SER-BagBoosting Trees 算法。

4 实验设计

4.1 实验数据说明

本节回归案例所使用的数据集为 3 个实际数据集，分别为 Boston Housing，Ozone，Algae。其中 Boston Housing，Ozone 均来自 UCI 机器学习库中，是经常被用于评价回归模型性能的典型数据集。Algae 是一个关于海藻繁殖数据集（原始数据可从 http://www.liacc.up.pt/%7Eltorgo/DataMiningWithR/处获取），共包含 340 个水样，删除两条包含过多缺失数据的水样后分为训练集和测试集。表 2 列出了这 3 个实际数据集的信息概要。

表 2 实际数据集基本信息概要

数据集	训练集	测试集	特征	输出
Boston Housing	406	100	11	1
Ozone	264	66	9	1
Algae	270	68	11	1

4.2 实验分析过程及结果

笔者选择了 CART，Boosting ，Bagging，Random Forest，SER-BagBoosting Trees 等算法对以上数据进行分析处理。对于如 Bagging 等集成学习算法，为克服一次结果的随机性，笔者采取了重复以上学习过程 50 次，取 50 次平均结果作为最终的预测输出的方式。表 3 列出了它们的预测精度比较。

由表 3 可知，SER-BagBoosting Trees 算法与其它算法相比，预测精度有一定的提高，尤其是要优于简单的 Bagging 和 Boosting 算法，且算法运行较稳定，在训练集和测试集上的预测精度相差不大。与 Random Forest 算法相比，在某些数据集上的运行效果要优于 Random Forest 算法，且集成所需要的基学习器的个数远远小于 Random Forest 算法构建时所需要的基学习器个数，即 SER-BagBoosting Trees 算法运行效率要高于 Random Forest 算法，特别是在大数据集上更为明显。

表 3 各算法在各数据集上的预测精度比较

数据集	学习方法	预测精度	
		训练集	测试集
Boston Housing	CART	0.819	0.746
	Boosting	0.887	0.859
	Bagging	0.864	0.838
	Random Forest	0.944	0.892
	SER-BagBoosting Trees	0.889	0.881
Ozone	CART	0.794	0.61
	Boosting	0.842	0.816
	Bagging	0.845	0.827
	Random Forest	0.829	0.813
	SER-BagBoosting Trees	0.872	0.856
Algae	CART	0.482	0.417
	Boosting	0.631	0.551
	Bagging	0.702	0.528
	Random Forest	0.685	0.545
	SER-BagBoosting Trees	0.643	0.572

结束语 本文提出的 SER-BagBoosting Trees 算法可以看作是一种选择性集成学习算法。它是一种综合了 Boosting 和 Bagging 算法各自特点并利用聚类技术和贪婪算法进行选择性集成的学习算法。从实践运行来看，它对一些数据的处理要优于简单的 Bagging 和 Boosting 算法，与 Random Forest 算法相比也毫不逊色。

参 考 文 献

[1] Zhou Z H, Wn J, Tang W. Ensembling neural networks: Many could be better than all[J]. Artificial Intelligence, 2002, 137(1/2): 239-263

[2] Breiman L. Bagging predictors[J]. Machine Learning, 1996, 24(2): 123-140

[3] Breiman L. Random forests[J]. Machine Learning, 2001, 45(1): 5-32

[4] Breiman L. Arcing the edge[J]. The Annals of Statistics, 1998, 26(3): 801-823

[5] Schapire R E, Freund Y, Bartlett P, et al. Boosting the margin: a new explanation for the effectiveness of voting methods[J]. The Annals of Statistics, 1998, 26(5): 1651-1686

[6] Dettling M. BagBoosting for tumor classification with gene expression data[J]. Bioinformatics, 2004(20): 3583-3593

（上接第 156 页）

研究力量投入到这个领域。

参 考 文 献

[1] 易芝，汪林林，王练. 基于关联规则相关性分析的 Web 个性化推荐研究[J]. 重庆邮电大学学报：自然科学版，2007，19(2)：234-237

[2] 纪良浩，王国胤，杨勇. 基于协作过滤的 Web 日志数据预处理研究[J]. 重庆邮电学院学报：自然科学版，2006，18(5)：646-649

[3] Pyle D. Data Preparation for Data Mining[M]. San Francisco, CA: Morgan Kaufmann Publishers Inc, 1999: 540

[4] Cooley R, Mobasher B, Srivastava J. Data Preparation for Mining World Wide Web Browsing Patterns [J]. Journal of Knowledge and Information Systems, 1999, 1(1): 32-57

[5] 代宇，刘宴兵，程瑶. 基于异步 Web Service 调用的 Web 应用程序研究[J]. 重庆邮电大学学报：自然科学版，2008，20(6)：746-748

[6] Marquardt C, Becker K, Ruiz D. A pre-processing tool for Web usage mining in the distance education domain[C]// Proceedings of the International Engineering and Applications Symposium (IDEAS'04)

[7] W. W. W. consortium. The Common Log File Format [EB/OL]. http://www.w3.org/Daemon/User/Config/Logging.htmlJHJcommon-logfile-format, 1995

[8] 费爱国，王新辉. 一种基于 Web 日志文件的信息挖掘方法[J]. 计算机应用，2004，24(6)：57-59

[9] Catledge L, Pitkow J. Characterizing Browsing Behaviors on the World Wide Web[J]. Computer Networks and ISDN Systems, 1995, 27(6)

[10] Chen M S, Park J S, Yu P S. Efficient Data Mining for Path Traversal Patterns[J]. IEEE Transaction on Knowledge and Data Engineering, 1998, 10(2): 209-221

[11] 朱秋云. 一种关联规则挖掘筛选算法设计[J]. 重庆工学院学报：自然科学版，2008，22(6)：115-117