

# Web 访问挖掘的预处理技术的研究

熊忠阳,周亚峰

(重庆大学 计算机学院,重庆 400044)

**摘要:** Web 日志挖掘就是运用数据挖掘技术从 Web 日志中发现和抽取信息的过程。数据预处理是 Web 日志挖掘的一个关键环节。对数据预处理的各个环节进行研究,并介绍各个环节中的一些特殊处理方法,根据对 Web 服务器日志数据格式的分析,对会话概念进行了形式化描述,然后在分析目前会话构造算法的基础上,提出了基于时间和引用的启发式方法来构造会话。

**关键词:** Web 挖掘; Web 日志挖掘; 数据预处理; 用户会话; 会话识别

**中图分类号:** TP311

**文献标识码:** A

**文章编号:** 1673-629X(2007)08-0011-04

## Research on Data Preprocessing Technology in Web Log Mining

XIONG Zhong-yang, ZHOU Ya-feng

(Department of Computer Science, Chongqing University, Chongqing 400044, China)

**Abstract:** Web log mining is a process that using data mining technology to find and extract information from Web log. Data preprocessing plays a key role in the process of Web log mining. This paper mainly researches all links of data preprocessing, introduces the solution of some especial problems in this process. By the analysis of Web server log format, give the formal descriptions of the concept of session. On the basis of analyzing the current session construction methods, mainly proposes the time-referrer-based heuristic method that can be used to construct sessions.

**Key words:** Web mining; Web log mining; data preprocessing; user session; session identification

## 0 引言

Web 挖掘是从 WWW 相关资源上抽取信息或知识的过程,它是将传统的数据挖掘的思想和方法应用于 Web,从 Web 文档和 Web 活动中抽取感兴趣的、潜在的、有用的模式或隐藏信息。

Web 日志挖掘是 Web 挖掘的一种,所有的网上行为(又称用户行为)的可记录性和数据量的快速增长促成此项应用的需求。Web 日志挖掘的对象主要是服务器上的日志信息,日志信息包括服务器日志、引用日志和客户端 Cookie<sup>[1]</sup>。

一般而言,Web 日志挖掘又可以分为数据预处理阶段、模式发现阶段和模式分析阶段。由于原始日志文件是简单的平面文本文件,包括了一些不完整的、冗余的、错误的数据,同时原始 Web 日志文件具有半结构化的特点,于是需要对原始日志文件进行处理,否则将影响挖掘的效果。

## 1 数据预处理前的预备知识

### 1.1 日志记录的内容

对目前大多数的 Web 日志挖掘来说,Web 服务器日志是主要数据来源,它记录了多个用户对该服务器进行访问时的页面请求信息<sup>[2]</sup>。目前日志文件以多种数据格式存储在 Web 服务器上。常见的日志数据格式是 W3C 联盟(World Wide Web Consortium)规定的常规日志格式(Common Log Format)和扩展日志格式(Extended Common Log Format),其结构如表 1 所示。

表中打黑点的部分是扩展型日志格式中添加的记录项,其中一些内容在实际应用中是用不到的,如 Cookie 和 URI 查询。用户 ID 是在被访问的页面需要进行用户认证时才有的,一般都为空;用户代理记录的是客户端使用的操作系统和浏览器的类型和版本。这里需要说明的是 URI 是一个比 URL 更通用的定义,而且前者包括后者。

### 1.2 几个概念

为保证数据挖掘的一致性,W3C 为此专门制定了一个适用于网络使用分析的网络数据项草案<sup>[3]</sup>。下面列出该文将用到的几种抽象数据类型:

收稿日期:2006-10-19

作者简介:熊忠阳(1962-),男,重庆人,教授,博士生导师,研究方向为并行数据挖掘技术与应用,信息网络与信息系统等。

表 1 Web 日志记录的主要信息

域	描述
日期(date)	用户请求页面的日期
时间(time)	用户请求页面的具体时间
客户 IP 地址(c-ip)	客户端主机的 IP 地址或 DNS 入口
用户名(cs-username)	客户端的用户名
·服务器名(s-computername)	服务器名称
服务器 IP 地址(s-ip)	服务器的 IP 地址
服务器端口(s-port)	服务器的端口号
方法(cs-method)	用户请求的方法
URI 资源(cs-uri-stem)	用户所请求的页面
URI 查询(cs-uri-query)	用户欲进行的查询
协议状态(sc-status)	返回 http 的状态标识
·发送字节数(sc-bytes)	服务器发送的字节数
·接收字节数(cs-bytes)	服务器收到的字节数
·所花时间(time-taken)	完成浏览所花费的时间
·协议版本(cs-version)	传输用的协议版本
·主机(cs-host)	服务器的操作系统
·用户代理(cs=User-Agent)	服务的提供者
·Cookie(cs=Cookie))	Cookie 标识号
·参照(cs=referrer))	用户浏览的上一页

1) 用户(User): 通过浏览器访问网页的个体, 用户可以通过在不同的地方通过不同的机器使用不同的代理访问网站。

2) 页面浏览(page view): 一个用户请求的页面, 一个页面中可能包括若干个框架、图片和脚本, 一次页面浏览代表一次用户行为。

3) 点击流(click-stream): 用户访问的一组连续的页面浏览的序列。

4) 用户会话(user session): 某个用户对整个网络进行访问时的一串顺序的页视图请求, 也就是单用户对多站点的访问。

5) 服务器会话: 在一个使用者会话中由对某个特殊网站进行访问的页面视图组成的一个集合, 也就是单个用户对单个站点访问。

6) 片段: 用户会话中一段有意义的点击流, 文中称为事务。

Web 日志挖掘的数据预处理的结果影响到挖掘算法产生的规则和模式, 因此预处理过程也是 Web 日志挖掘质量保证的关键, 它包括数据清理、用户识别、会话识别、路径补充、事务识别五个阶段。

## 2 数据预处理过程

### 2.1 数据清理

数据清理是根据需求对原始日志文件进行处理, 删除 Web 服务器日志中与挖掘算法无关的数据, 这些数据有以下几类:

1) 图片、视频等非用户显式请求的逻辑单位。当用户请求一个页面时, 与这个页面有关的图片、视频等信息会自动下载, 并记录在日志文件中, 但是对用户来说, 一般显式请求的是后缀为 .html 和 .htm 的文本页

面, 所以应该将日志中文件的后缀为 gif, jpg, jpeg 等的记录删除。这可以通过对 URI 域的处理实现。

2) Web Robot 的浏览日志记录。由于搜索引擎或其它一些自动代理的存在, 日志中存在大量由它们发出的请求, 这些项并不能反应用户的访问行为, 应该清除掉。这可以通过检查日志中每项的代理域, 许多 robot 的代理值与通常的浏览器不一样, 可以通过字符串匹配的方式删除这些日志记录。

3) 一些状态域错误的日志。状态域代码为 200~299 的一般指示成功响应; 300~599 指示各种不同类别的错误, 于是可以通过对协议状态域的检查, 凡是代码值大于 299 的日志记录都应该删除。

4) 由 GET 以外的方式完成的服务。常见的请求方法有 GET, POST 和 HEAD, 但是只有 GET 方法反应了用户的访问行为, 所以应该清除 GET 以外的方式, 可以通过检查方法域进行清除。

5) 后缀为 cgi, js, JS 的脚本文件。这些文件对后面的分析处理不造成任何影响, 应该删除。

### 2.2 用户识别

用户就是一个独立的个体, 它通过一个浏览器访问一个或多个 Web 站点。但由于本地 cache、代理服务、防火墙的存在, 使得用户识别比较困难。如: 通过代理服务器上上网的用户在日志文件中的 IP 地址相同; 由于防火墙的存在, 在服务器日志中多个用户的 IP 地址相同<sup>[4]</sup>。为此, 目前的用户识别都是采用以下三条启发式原则:

1) 如果用户的 IP 地址不同则认为是不同的用户。

2) 如果 IP 地址相同, 但浏览器软件或操作系统不同(用户代理域), 则认为是不同的用户。

3) 如果 IP 地址相同, 用户使用的操作系统和浏览器软件也相同, 那么根据网站的拓扑结构对用户进行识别, 如果用户请求的页面不能从已访问的任何页面到达, 则判断这是一个新的用户。

### 2.3 会话识别

在跨越时间区段较长的 Web 服务器日志中, 用户有可能多次访问了该站点, 会话识别的目的就是将属于同一个用户的同一次访问请求识别出来。

定义 1 Web 服务器日志可以看成是按照时间戳排序的集合  $L = \{l_1, l_2, \dots, l_i, \dots, l_n\} (1 \leq i \leq n)$ 。| $L$ | =  $n$  表示日志集合  $L$  包含的日志数目, 即 Web 服务器日志中记录的页面请求数。集合  $L$  的每个元素  $l$  具有以下性质<sup>[5]</sup>:

1)  $l_i = \{\text{userid}, \text{url}, \text{time}, \text{refer}\} (1 \leq i \leq n)$  这里的 userid 是经过用户识别后赋予每个用户唯一的标识符。

2)  $l_i.time < l_j.time (l \leq i < j \leq n)$ 。

定义2 会话  $r$  表示为  $r = \{userid, (< l'_1.url, l'_1.time, l'_1.refer >, \dots, < l'_m.url, l'_m.time, l'_m.refer >)\}$ , 其中  $1 \leq k < j \leq m$ , 且  $l'_k.userid = userid, l'_k.time < l'_j.time$ 。length( $r$ ) 称为会话  $r$  的长度, 即会话  $r$  所包含的页面请求数目, 在这里  $length(r) = m$ 。

定义3 所有的会话所构成的集合被称为会话集  $R = \{r_1, r_2, \dots, r_i, \dots, r_n\}$ 。| $R$ | =  $s$  表示会话集中包含的会话个数。

会话的划分方法很多, 有的依据时间, 有的依据站点拓扑结构。人们常用以下三个算法划分会话, 前2个基于时间, 后一个基于网站结构<sup>[6]</sup>:

1) 给用户一次会话时间规定一个阈值  $T_1$  ( $T_1 = 25.5\text{min}$ , 很多商业产品采取  $30\text{min}$  作为阈值, 而 L. Catledge 和 J. PitKow 试验所得数据指出  $25.5\text{min}$  更为合适), 如果超过这个阈值  $T_1$  则认为新会话的开始。在相同用户的前提下, 设  $t_0$  为会话初始页的时间戳, 一个 URL 请求的时间戳  $t$  满足  $t - t_0 \leq T_1$ , 则这个请求加入到本次会话, 第一个满足  $t - t_0 > T_1$  的页面成为一个新的会话的初始页面。

2) 给用户一个页面的请求时间规定一个阈值  $T_2$  ( $T_2 = 10\text{min}$ ), 如果2个连续请求的时间间隔超过了这个阈值  $T_2$ , 则认为是一个新会话的开始。设  $t_0$  为当前会话最后一个请求的时间戳, 如果下一个请求的时间戳  $t$  满足  $t - t_0 \leq T_2$ , 则加入当前会话, 否则成为一个新的会话的初始页面。

3) 由于 Web 浏览大多是通过点击超链接得到, 如果同一个用户连续发出相邻的两个页面请求  $p$  (其中  $p$  属于会话  $S$ ) 和  $q$ ,  $t_p$  和  $t_q$  分别代表页面请求  $p$  和  $q$  的时间戳 ( $t_p < t_q$ )。如果页面请求  $q$  的引用页面增加在会话  $S$  中出现过, 那么  $q$  就属于会话  $S$ ; 或者  $q$  的引用页面为空并且  $t_q - t_p \leq \Delta$  ( $\Delta$  为时间延迟, 这里取  $10\text{s}$ ), 那么页面请求  $q$  属于会话  $S$ 。

上述3) 中加入时间限制, 是因为超级链接并非都是由其参考页得到, 用户可能通过点击浏览器上的“BACK”按钮, 回溯到某个曾经浏览过的页面得到该页, 可以把时间间隔较短的请求划分到一个会话中。

目前的一些研究者提出会话构造算法大多采用上述三个算法中的其中一个来进行构造, 这样得出的会话集合必定与真实会话之间存在很大差距。比如: 单纯采用时间阈值  $T_1$ , 实际存在用户在  $25.5\text{min}$  阈值内同时进行着两个以上的会话 (比如同时打开几个浏览器窗口, 在一个窗口内容下载过程中, 浏览另一个窗口的内容); 单纯采用相邻请求页面间的引用关系, 可能存

在用户浏览器一直开启, 下次访问时直接在先前已经打开的页面上进行访问的情况, 这样就会把时间跨度很大的用户多次访问划到一次会话当中。于是文中提出一种基于三者集合的思想的算法, 这样就可以成功避免以上的问题。主要算法过程如下:

(1) 构造候选会话集合。

算法: 构造候选会话集合 candidate\_session( $L, Rc$ );

输入: Web 服务器日志集合  $L$ , 一次会话时间阈值  $T_1 = 25.5\text{min}$ , 连续两个页面请求的时间阈值  $T_2 = 10\text{min}$ ;

输出: 候选会话集合  $Rc = \{\}$ ;

Procedure candidate\_session( $L, Rc$ )

```
{
    i := 1;
    while i ≤ |L| do
    {
        t := {li.userid, (< li.url, li.time, li.refer >)};
        j := i;
        while li+1.userid = li.userid do
        {
            if ((li+1.time - lj.time) ≤ T1) ∧ ((lj+1.time - li.time) ≤ T2)
            {
                t := t ∪ < li+1.url, li+1.time, li+1.refer >;
            }
            else
            {
                Rc := Rc ∪ t;
                t := {li+1.userid, (< li+1.url, li+1.time, li+1.refer >)};
            }
            i := i + 1;
        }
        Rc := Rc ∪ t;
        i := i + 1;
        j := i;
    }
}
```

(2) 生成会话集合  $R$ 。

算法: 生成会话集合 generate\_session( $Rc, R$ );

输入: 候选会话集合  $Rc$ , 页面请求延迟时间  $\Delta = 10\text{s}$ ;

输出: 会话集合  $R = \{\}$ ;

Procedure generate\_session( $Rc, R$ );

```
{
    i := 1;
    while i ≤ |Rc| do
    {
```

```
j := 0;
r := {rcj.userid, (< l1rc.url, l1rc.time, l1rc.refer >)};
j := j + 1;
while (j ≤ length(rc)) do
{
  if (lj+1rc.refer 与任一 lkrc.url(1 ≤ k ≤ j) 相同) or
  (lj+1rc.refer = null and lj+1rc.time - ljrc.time ≤ Δ)
  then
  {
    r := r ∪ < lj+1rc.url, lj+1rc.time, lj+1rc.refer >;
  }
else
{
  R := R ∪ r;
  r := {rcj.userid, (< lj+1rc.url, lj+1rc.time, lj+1rc.refer >)};
}
j := j + 1;
}
i := i + 1;
}
```

2.4 路径补充

用户浏览网页时,通过按下浏览器上的“后退”按钮得到的页面是从本地缓冲区中得到的,在日志文件中是没有记录的,从而导致该页与用户上一次请求的页面之间没有超链接信息。在这种情况下,可以根据网站的拓扑结构,把用户的访问路径补充完整,如果在用户的历史访问记录中有多个页面都包含与当前请求页的链接,则将请求时间最接近当前页的页面作为当前请求的来源。在路径补充中遵循的一个原则就是在用户的一次会话的页面集中,除了起始页面外,每个页面同它上一个页面存在链接关系。

2.5 事务识别

事务识别也叫片段识别,目的是找出会话中有意义的访问路径<sup>[7]</sup>。一般来说用户从缓冲区中取得的网页不是用户真正想浏览的网页,用户只是将其作为桥梁来访问其它的网页,并没有实在的意义。目前事务识别采用的方法是识别最大向前引用路径,一个最大向前引用路径就相当于一个事务。向后引用意味着一个用户再次请求其浏览过的页面(如用户按下“返回”按钮),当一个向后引用发生时,说明向前引用终止,则得到的向前引用的路径即是一个最大向前引用,或者当这个用户会话结束时,也获得一个最大向前引用。

3 试验数据

下面取重庆大学民主湖论坛所在服务器上的日志记录中的一部分日志测试上面的用户识别算法。原始

的访问日志记录如表 2 所示(为简化表格,A 代表“Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)”,B 代表“Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.0)”,这里 Windows NT 5.0 代表的是 Win2000 操作系统,Windows NT 5.1 代表的是 WinXP 操作系统。对网页的 URI 用英文字母和数字代替,省掉 Time/Date 域中的时区 +0800,代表中国)。

表 2 原始日志记录

序号	IP Address	Time/Date	Method/URI	Referer	Agent
1	219.221.121.47	[22/July/2006:09:10:21]	"Get index.html"	-	A
2	219.221.121.47	[22/July/2006:09:10:30]	"Get A.html"	index.html	A
3	219.221.121.47	[22/July/2006:09:10:45]	"Get K.html"	-	A
4	219.221.121.47	[22/July/2006:09:10:48]	"Get E.html"	A.html	A
5	219.221.121.47	[22/July/2006:09:11:06]	"Get index.html"	-	B
6	219.221.121.47	[22/July/2006:09:11:21]	"Get A.html"	index.html	B
7	219.221.121.47	[22/July/2006:09:11:43]	"Get Q.html"	K.html	A
8	219.221.121.47	[22/July/2006:09:12:21]	"Get B.html"	index.html	B
9	219.221.121.47	[22/July/2006:09:12:49]	"GET N.html"	E.html	A
10	219.221.121.47	[22/July/2006:09:14:13]	"Get I.html"	B.html	B
11	219.221.121.47	[22/July/2006:09:14:27]	"Get F.html"	A.html	A
12	219.221.121.47	[22/July/2006:10:16:53]	"Get index.html"	-	A
13	219.221.121.47	[22/July/2006:10:17:24]	"Get C.html"	index.html	A

将上面讨论的 Web 日志预处理的几个过程中的相关算法作用于表 1 中的日志记录,得到的结果如表 3 所示(由于 IP Address 相同,这里省略)。

表 3 经过数据预处理后的用户会话记录

序号	IP Address/Agent	Time/Date	Request URI	Referer URI
1	"Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)"	[22/July/2006:09:10:21]	index.html	-
		[22/July/2006:09:10:30]	A.html	index.html
		[22/July/2006:09:10:48]	E.html	A.html
		[22/July/2006:09:12:49]	N.html	E.html
		[22/July/2006:09:14:27]	F.html	N.html
2	"Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)"	[22/July/2006:10:16:53]	index.html	-
		[22/July/2006:10:17:24]	C.html	index.html
3	"Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1)"	[22/July/2006:09:11:06]	index.html	-
		[22/July/2006:09:11:21]	A.html	index.html
		[22/July/2006:09:12:21]	B.html	A.html
		[22/July/2006:09:14:13]	L.html	B.html
4	"Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.0)"	[22/July/2006:09:10:45]	K.html	-
		[22/July/2006:09:11:43]	Q.html	K.html

4 结束语

文中主要对 Web 日志挖掘的数据预处理部分进行研究,分析了数据预处理的各个阶段需要解决的问题和采用的解决方法。针对数据预处理中的关键阶段一会话识别,提出了一个基于会话时间、页面访问时间

(下转第 18 页)

mpirun -hosts 2 hostname1 hostname2 E: \ MPI \ RunFile \ MPIHohai. exe 即在 hostname1 和 hostname2 两台计算机上运行并行程序,每个计算机开一个进程

mpirun -hosts 2 hostname1 1 hostname2 2 E: \ MPI \ RunFile \ MPIHohai. exe 即在 hostname1 和 hostname2 两台计算机上运行并行程序,hostname1 上开一个进程,hostname2 上开两个进程。



图 6 设置系统环境变量

### 3.4 以配置文件的形式执行并行程序

在前一节中曾提到若要多台计算同时运行并行计算程序,则并行程序在所有计算机上的路径和文件名必须一致才可运行。然而,可以通过设置配置文件来执行并行程序,这样并行程序在不同计算机上的路径和文件名可以不同。

若配置文件 MPI. cfg 格式如下:

```
-hosts 1 hostname1 1 E: \ MPI \ RunFile \ abc1. exe
```

```
-hosts 1 hostname2 2 F: \ abc2. exe
```

```
-hosts 1 hostname3 4 G: \ MPI \ RunFile \ abc3. exe
```

以上配置文件说明并行可执行程序在 hostname1 上位于 E: \ MPI \ RunFile \ abc1. exe,以一个进程运行;并行可执行程序在 hostname2 上位于 F: \ abc2. exe,以两个进程运行;并行可执行程序在 hostname3 上位于 G: \ MPI \ RunFile \ abc1. exe,以四个进程运行;

在 DOS 命令窗口里输入:

```
mpirun -configfile MPI. cfg
```

即可在 hostname1,hostname2 和 hostname3 三台计算机上同时运行可执行程序。尽管三台计算上的并行程序的路径和文件名各不相同,但这三个可执行程序必须是完全一样的程序。

## 4 结 论

文中详细介绍了 WindowsXP 平台下 MPICH2 的 Fortran90 并行编译环境的安装、配置及并行程序的编译、连接和运行方法,并进一步介绍了机群环境下运行并行计算程序的方法,为现有 Fortran 串程序的并行化提供了编译环境。

### 参考文献:

- [1] 曾庆华,陈天麟.可扩展并行计算机系统结构和发展现状[J].计算机科学,2003,30(9):158-161.
- [2] 都志辉.高性能计算并行编程技术-MPI并行程序设计[M].北京:清华大学出版社,2001.
- [3] 曾志辉.Linux环境下MPI并行编程与算法实现研究[J].航空计算技术,2004,34(2):61-64.
- [4] 李俊照,罗家融.基于linux集群的并行计算[J].计算机测量与控制,2004,12(11):1064-1066.
- [5] 许丽华,刘 森.MPI并行编程环境的研究[J].应用技术,2003(4):28-31.

(上接第 14 页)

和网站拓扑结构三个要素结合的会话构造算法,从而使会话识别出来的结果更接近于用户的真实会话。

### 参考文献:

- [1] Han Jiawei, Kamber M. Data Mining[M]. Beijing: Higher Education Press, 2000.
- [2] Serivastava J, Cooley R, Deshpande M, et al. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data[J]. ACM SIGKDD Explorations, 2000, 1(2): 12-23.
- [3] Spiliopoulou M, Mobasher B, Berendt B, et al. A framework for the evaluation of session reconstruction heuristics in Web usage analysis[J]. Inform Journal on Computing, 2003, 15(5): 171-179.

- [4] Baglioni M, Ferrara U, Romei A, et al. Preprocessing and mining Weblog data for Web personalization[C]//Proceedings of 8th Natl' conf of the Italian Association for Artificial Intelligence. Pisa, Italy: [s. n.], 2003.
- [5] 赵 伟,何丕廉,陈 峡,等. Web 日志挖掘中的数据预处理技术研究[J]. 计算机应用, 2003, 23(5): 62-66.
- [6] Wang Xidong, Ouyang Yiming, Hu Xuegang, et al. Discovery of User Frequent Access Patterns on Web Usage Mining [C]//In: The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings. [s. l.]: IEEE, 2003.
- [7] 张 娥,郑斐峰,冯耕中,等. Web 日志数据挖掘的数据预处理方法研究[J]. 计算机应用研究, 2004(2): 58-60.