

# Web 使用挖掘的数据采集技术探究

邵兰洁<sup>1</sup>, 李光忠<sup>2</sup>

(1. 北京化工大学 北方学院信息学院, 河北 廊坊 065201;

2. 山东农业大学 信息科学与工程学院, 山东 泰安 271018)

**摘 要:**如何准确、及时、全面地采集用户使用数据是 Web 使用挖掘的重要前提和基础。基于 Web 的基本结构, Web 使用挖掘的数据源可以从 Web 服务器端、应用服务器端、代理服务器端和客户端进行采集。文中分析了传统的基于 Web 日志进行 Web 使用挖掘所面临的问题, 讨论了建立在用户浏览行为基础上的客户端数据采集技术, 重点讨论了其中的 Java Applet 技术。通过 Java Applet 技术可以获取客户端 IP, 可以自动完成用户浏览信息的准确采集, 可以广泛用于各类网站的个性化和智能化服务、站点结构改进、商业智能等。

**关键词:**数据采集; Web 使用挖掘; Web 日志; Java Applet

**中图分类号:**TP311

**文献标识码:**A

**文章编号:**1673-629X(2010)03-0225-05

## Research on Techniques of Data Collecting for Web Usage Mining

SHAO Lan-jie<sup>1</sup>, LI Guang-zhong<sup>2</sup>

(1. Information Institute, North College of Beijing University of Chemical Technology,  
Langfang 065201, China;

2. College of Information Science and Engineering, Shandong Agriculture University,  
Tai'an 271018, China)

**Abstract:**How to collect users' data accurately and quickly and ensure data integrity is an important precondition and foundation for Web usage mining research. Based on the Web structure, the data source of Web usage mining can be collected from Web server, application server, agent server and client. In this paper, the problems facing traditional Web usage mining based on the Web log are analysed, the data collection techniques of client are discussed which is based on the users' browsing behaviours, and the Java Applet technique is much emphasized, which can help get the IP address of client, automatically complete the accurate collection of users' browsing information, can be widely used for the Web sites' personal and intelligent service, for the improvement of Web structure, and for the business intelligence, etc.

**Key words:**data collecting; Web usage mining; Web Log; Java Applet

## 0 引 言

当前, WWW 正在深度和广度方面飞速地发展着, 也正在前所未有地改变着我们的生活。WWW 上的一些主要工作, 例如 Web 站点设计、Web 服务设计、Web 站点的导航设计、电子商务等工作正变得越来越复杂和越来越繁重。从站点经营方来说, 他们需要好的自动辅助设计工具, 可以根据群体用户的访问兴趣、访问频度、访问时间动态地调整页面结构, 改进服务, 以更

好地满足访问者的需求。从访问者来说, 他们希望看到的是个性化的页面, 希望得到更好的满足各自需求的服务。这种需求从某种意义上说, 访问者本身也未必清楚。欲解决这两方面需求, 一个有力的工具就是 Web 使用挖掘。WWW 的数据挖掘目前有很大一部分集中在 Web 使用挖掘上。

所谓 Web 使用挖掘, 是指从 Web 服务器端记录的用户访问日志和/或从用户的浏览信息中抽取感兴趣的模式的过程。一般分为 4 个阶段: 数据采集、数据预处理、模式发现、模式分析<sup>[1,2]</sup>。其中数据采集是 Web 使用挖掘的一个关键步骤, 是 Web 使用挖掘的重要前提, 数据采集所获取的源数据质量直接关系到最终挖掘结果的质量。

收稿日期: 2009-06-08; 修回日期: 2009-09-28

基金项目: 国家“十一五”计划项目(FIB070335-B8-08)

作者简介: 邵兰洁(1973-), 女, 山东菏泽人, 讲师, 硕士, 研究方向为数据库技术、计算机网络技术、信号与信息处理。

## 1 Web 日志挖掘面临的问题

Web 的基本结构是:客户端—代理服务器—Web 服务器—应用服务器,如图 1 所示。所以 Web 使用挖掘的数据源可以从 Web 服务器端、应用服务器端、代理服务器端和客户端进行采集,传统的 Web 使用挖掘的数据采集大都从 Web 服务器日志而来<sup>[3~5]</sup>。

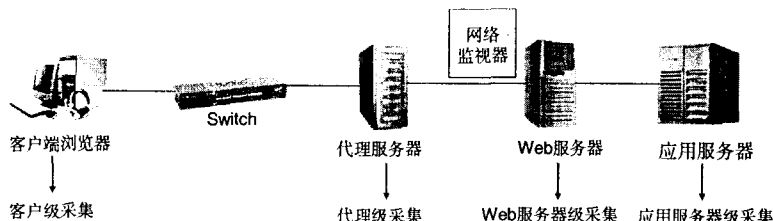


图 1 4 级数据采集技术

一般的 Web 服务器日志记录形式如下:

219.233.102.26 - - [3/Aug/2009:10:35:33 + 0800] "GET/xinxi/index.html HTTP/1.1" 200 631 "http://www.ncbuct.com/" "Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; MyIE2)"

每当站点上的网页被访问一次,Web 服务器就在日志文件中增加一条相应的记录。这些记录数据反映了多个用户(可能同时)对 Web 站点(单站点)的访问行为。

通过对 Web 日志文件记录方式的研究,已经知道,用户对 Web 服务器的一次页面请求(即一次点击),往往会在服务器的日志文件中产生多条记录。表面上看,用户对浏览器的一次点击会发送给服务器一个请求,调用网站的一个页面,但同时也打开了该网页所涉及的全部对象,而每一个对象都在日志文件里产生一条记录。所以,基于 Web 日志的使用挖掘就必须通过数据净化(Data Cleaning)操作对这些海量数据进行数据预处理,剩下的可用数据约为 5%~10%<sup>[2]</sup>。

Web 日志一般缺乏用户 ID,因为只有当请求文件需要认证时,用户一栏才有数据。并且 Web 环境中代理服务器和防火墙的存在,使得不同的用户请求在 Web 服务器的日志中记录的都是代理服务器或防火墙的 IP 地址,所以单靠 Web 服务器日志数据很难准确确定用户。

Web 环境中存在多级别的缓存,如用户本地缓存和代理服务器缓存。用户浏览缓存的页面不在服务器端日志上记录,导致了 Web 日志文件中对用户页面浏览记录的不完整性。在有的场所,缓存能提供 20%~50% 的请求服务<sup>[6]</sup>,这也就意味着服务器日志丢失了很多的用户浏览行为记录。

在 Web 使用挖掘中,用户在一个页面上浏览的时

间是一个很重要的信息,它可以看出用户对该页面的兴趣程度。基于 Web 日志计算用户在一个页面的逗留时间时,一般用下一个 HTTP 请求的 time 域减去当前 HTTP 请求的 time 域得到,但由于网络传输和服务器响应时间等原因导致通过这种方法计算浏览时间不准确;当用户查看缓存中的网页时也会把浏览缓存内网页的时间累加到上一个 HTTP 请求返回的页面上;当用户离开网站时,浏览器并不发生 HTTP 请求,导致最后一个页面的查看时间也就无法得到等。

此外,搜索引擎、镜像服务器以及 Flash 页面的大量使用等,进一步影响了基于 Web 日志文件的 Web 使用挖掘的数据采集的准确性。

总之,Web 服务器日志中的网站使用数据并不完全可靠,未采集到所有访问页面、采集时间不够准确、浏览用户的确定不够准确。因此,用服务器端记录的数据进行 Web 使用模式挖掘也是不完全可靠的。Web 服务器日志在应用于传统的 Web 使用挖掘之前必须经过预处理,其过程包括:依赖于域的数据净化(Data Cleaning)、用户识别(User Identification)、会话识别(Session Identification)、路径补充(Path Completion)、事务识别(Transaction Identification)<sup>[7,8]</sup>,以获得相对完整和准确的用户使用数据。

## 2 客户端数据采集

客户端的数据采集技术包括远程代理(如 Java Applet 或 Java Script)、Plug-in、网页跟踪帧和修改已有浏览器等。客户端的数据采集比直接从 Web 日志文件采集用户使用数据更具有优越性,因为它是建立在用户浏览行为源上的,可准确地捕获用户的浏览行为,能准确地确定浏览用户,用户的浏览路径和浏览时间都可精确地测量,避免了用户识别、会话识别、路径补充等繁琐的数据预处理过程。但它需要用户的许可,有可能会触及到用户的隐私。

### (1) Java Applet 技术。

在每个需要跟踪的网页主体部分的开始添加 Java Applet 程序代码,执行跟踪任务。当用户刚来访问 Web 服务器时把 Applet 下载到客户的浏览器上运行。每当页面切换时,Applet 会发送 URL 和浏览时间到服务器。这种方法的不足之处是客户端浏览器需要下载并安装 Sun JVM 插件,且使 Java Applet 生效,打开 Java Applet 的允许开关。另外,Applet 在第一次被下载执行时可能会花费一些时间,下载时间受连接速度、网络阻塞、下载的 Java 类的数目等因素影响。

## (2) Java Script 技术。

Java Script 是在客户端直接编译执行的脚本语言。可在网页浏览开始和退出时的相应句柄(Onload 和 Onunload)中执行事件处理,把用户浏览网页的 URL 地址和相应的浏览时间发送到服务器。但是,应当注意:它只有在用户点击超链接并转到其它页面使该页面后退的时间才能得到,如果用户是点击浏览器上的“前进”或“后退”等按钮,Java 脚本由于没有相应的句柄而不能得到该页面的退出时间给服务器。

## (3) Plug-in 技术。

客户端以 Plug-in 方式嵌入监控程序,Plug-in 以后台操作方式随时跟踪网页的浏览情况。浏览器可以在一个页面上或同时在几个打开的窗口上用 New 加载同一插件的多个实例。当用户离开实例所在的页面或关闭窗口时,用 Destroy 删除插件的实例,并把相应的浏览信息以 Socket 方式送到服务器端。它的缺点是浏览器要事先安装 Plug-in。

## (4) 网页跟踪帧技术。

在一个浏览页面内以 0% 的隐含帧(跟踪帧)追踪另一个 100% 的帧内的浏览信息,并随时将客户端浏览信息以 WinSock TCP 的方式传到服务器。为了不延迟页面请求效率,将针对服务器的操作代码分离出来,放入一个独立的 PHP 文件,如 Server.php。在主页面中添加如下代码:

```
<iframe name="pagename"src="server.php"
width="0"height="0"></iframe>
```

<!-- 这里将 iframe 的高和宽设为 0 实现隐藏 -->

即在主页面中加入一个隐藏的内嵌页面(inner frame, IFrame),将 Server.php 放入内嵌页面打开,其它内容仍放在主页面。内嵌页面的下载不会影响主页面的下载,这就提高了主页面请求效率。

## (5) 修改浏览器技术。

直接采用增强跟踪功能的浏览器。目前,NetZerо、Alexa、AllAdvantage 就是这样做的,修改过源代码的浏览器能采集某个用户在多个网站上浏览的数据。

# 3 客户端数据采集的 Java Applet 实现

Java Applet<sup>[9,10]</sup>是 Java 的一种程序模式,这种程序可以嵌入到 Web 页面中,当用户浏览器在下载 HTML 文件时在用户机上执行。Java Applet 提供了 4 个可以自动调用的方法,可以通过这 4 个方法完成用户浏览信息的采集。

init()方法:当一个 WWW 浏览器下载一个含有 Applet 的页面时,首先执行的方法就是 init()方法。这一方法的任务是进行 Applet 初始化。它将产生一个

Applet 主类,对 Applet 自身进行初始化,然后启动 Applet,开始运行程序。

start()方法:当一个含有 Applet 的页面第一次被载入的时候,start 方法紧接着 init()方法被 WWW 浏览器调用。当用户离开该页面去浏览其它页面后,如果再返回该页面,则 WWW 浏览器直接执行 start()方法,重新载入 Applet,而不去执行 init()方法。

stop()方法:当用户离开某个包含 Applet 的页面去浏览其它页面时,WWW 浏览器会调用这个方法,以停止 Applet 的运行,但 Applet 并没有结束,只是处于一种休息状态。当用户再次进入到该页面时,WWW 浏览器只需要执行 start()方法,而不必再执行 init()方法。

destroy()方法:当用户退出浏览包含 Applet 的页面时,这个方法会被调用以结束程序。

图 2 显示了 Java Applet 的生命周期。

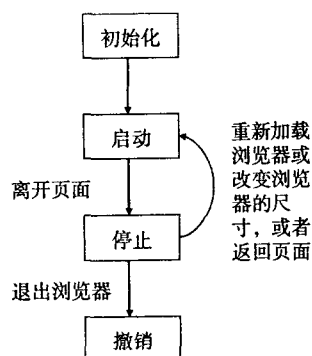


图 2 Java Applet 的生命周期

在 Web 站点的每个页面中都加上如下代码:

```
<html>
.....
<body>
<applet code="BrowseDetails.class">!-- 用户浏览信息客户端数据采集 Java Applet -->
name="BrowseDetails" width="1" height="1">
<!-- 在这里设宽、高为 1 是为了让用户在浏览页面时不可见此 Applet -->
<param name="URLName" value="news.html">!-- 传递给 BrowseDetails 页面 URL -->
</applet>
.....
</body>
</html>
```

当用户浏览网站页面时就会把 BrowseDetails.class 这样一个 Java Applet 下载到客户端的浏览器上运行。BrowseDetails.class 的源程序代码大致如下:

```
public class BrowseDetails extends Applet{
long StartTime,EndTime; //分别记录用户在一次浏览中的浏览
```

```

开始时间和结束时间
long Time; //记录用户的浏览时长
String userIP; //记录用户 IP 地址
Socket sock; //声明 Socket 类
String message;
public void init(){}
public void start(){
    StartTime = System.currentTimeMillis(); //当页面完全在载时获取
    页面浏览开始时间
}
public void stop(){
    EndTime = System.currentTimeMillis(); //当页面切换时获取页面
    浏览结束时间
    Time = EndTime - StartTime; //获取页面净浏览时间
    String BrowsingTime = java.lang.Long.toString(Time);
    String PageName = this.getParameter("URLName"); //获取页面
    URL
    try{
        UserIP = InetAddress.getLocalHost().toString(); //获取用户 IP
        地址
        String mes = UserIP + " " + PageName + " " + BrowsingTime
        //与 Servlet 服务器建立 Socket 连接
        sock = new Socket(this.getCodeBase().getHost(), Port);
        //打开 socket 输出流, 将页面 URL 和浏览时间发给服务器
        DataOutputStream out = new DataOutputStream(sock.getOutputStream());
        //把要发送的信息编码后保存在 string 字符串变量中
        message = URLEncoder.encode(mes, "UTF-8");
        //向服务器发送 POST 请求, 格式为 POST filename HTTP/1.0
        out.writeBytes("POST/netapp/servlet/acceptBrowseDetails " +
            "HTTP/1.0\r\n");
        //发送内容类型, 最后一定要有换行符
        out.writeBytes("Content-type: application/octet-stream\r\n");
        out.writeBytes("Content-length:" + message.length() + "\r\n");
        //发送请求的信息的长度
        out.writeBytes("\r\n"); //单独发送一空行, 表示发送头信息
        结束
        out.writeBytes(message); //发送信息的主体部分
        out.writeBytes("\r\n");
        out.close();
    }
    catch(UnknownHostException e) {...} //处理异常
    catch(IOException e) {...}
}
public void destroy(){}
}

```

由 Java Applet 采集的客户端用户浏览信息: 客户端 IP、访问页面和访问时间等, 交给后台类似 CGI 的

Servlet 程序来写入到服务器。

Servlet 是对支持 Java 的服务器的一般扩充, 它最常见的用途是增强和扩展 Web 服务器的功能, 增加 Web 服务器的互动性。Servlet 小应用程序也是用 Java 语言编写的在服务器一端运行的程序, 它继承了 Java 语言的所有优点, 如安全性、健壮性、可移植性等。并且, Servlet 小应用程序是与任何协议无关的, 即对传递它的协议没有任何限制。同时, 它也是与平台无关的。Servlet 小应用程序可以在 HTML 文档中被调用, 即它可以被远程启动。一个 Servlet 小应用程序被客户端发送的一个 HTTP 请求激活后, 它将一直运行于后台, 等待以后的客户端请求。每个客户端请求将生成一个新的线程, 而不是一个完整的进程, 多个客户能够在同一个 Servlet 进程中同时得到 Servlet 小应用程序的服务。它提供的强有力且高效率的技术正在逐步取代目前的 CGI 技术。

服务器端 Servlet 小应用程序的源代码 acceptBrowseDetails.java 大致如下:

```

public class acceptBrowseDetails extends HttpServlet{
    String Mes;
    //初始化 servlet 程序, 确保 servlet 在使用 service() 方法接受网页
    请求时, 已经完全加载结束, 否则抛出例外
    public void init(ServletConfig config) throws ServletException{
        super.init(config);
    }
    public void service(HttpServletRequest rq, HttpServletResponse rp)
        //从网页上接受请求
        throws ServletException, IOException{
        agentIP = rq.getRemoteAddr();
        File file = new File("serverlog.txt"); //客户端用户浏览信息日志
        文件
        RandomAccessFile raf;
        ServletInputStream in = rq.getInputStream(); //从 applet 获得的
        信息
        int len = rq.getContentLength();
        byte b[] = new byte[len]; //定义 byte 数组, 长度为内容的长度
        in.readLine(b, 0, len); //一行一行读取发送来的数据
        String message = new String(b); //将读取来的数据转换为字符串
        message = URLDecoder.decode(message, "UTF-8"); //将读取
        来的数据使用 UTF-8 格式解码
        if(! file.exists()) //如果文件不存在, 创建文件实例, 并写入从
        applet 获得的信息
        raf = new RandomAccessFile(file, "rw");
        raf.seek(0);
        raf.writeBytes(agentIP + " " + message);
        raf.close();
    }
    else //如果文件存在, 创建文件实例, 先读取原文件内容, 再写
    入从 applet 获得的信息

```

```

raf = new RandomAccessFile(file, "rw");
raf.seek(raf.length());
raf.writeBytes(agentIP + " " + message);
raf.writeBytes("\r\n");
raf.close();
}

public String getServletInfo()//获得 servlet 的信息
{ return "This is acceptServlet."; }
}

```

另外,利用 java.net 包中提供的 Socket 类和 ServerSocket 类,将其分别应用于客户机端的 Java Applet 小应用程序和服务端端的 Java 应用程序中,即可在两者之间(程序之间)建立套字连接。其中,Socket 类实现了客户端的通信功能,ServerSocket 类实现了服务器端的通信功能。当客户机和服务器连通后,它们之间就建立了一种双向通信模式。由 Java Applet 采集的客户端用户浏览信息,通过在客户机端的 Java Applet 小应用程序和服务端端的 Java 应用程序之间建立的套字连接就可以进行数据通信了。

文中采用 Java Applet 软件代理方法采集客户端数据,其突出优点是:

(1) 可以获取客户端 IP。如果客户端通过代理服务上网,我们采用客户端 IP 与代理 IP 的组合来进一步识别用户。

(2) 可以自动完成浏览信息采集。嵌入在 HTML 网页中的 Java Applet 小应用程序在网页被下载时,自动被下载到客户端并自动执行,采集用户浏览信息。在数据采集过程中不需要用户的显式参与,不影响用户的正常浏览。用户在浏览的时候,系统在客户端和服务端自动跟踪、记录用户的浏览行为,如浏览页面名称、开始浏览时间、浏览时长、浏览路径。信息采集完全由浏览器和 Web 服务器自动完成,不需要用户额外提供任何数据,用户不会产生任何不适感,不会影响用户正常的浏览行为。

(3) 采集到的用户浏览信息准确。由于用户丝毫不感觉不到数据采集对他的影响,他完全可以按照他自己的意愿进行浏览,不必为了完成额外的工作而多费脑筋,可以一心一意地进行浏览。而这一点正是个性化服务所必需的。

#### 4 结束语

在 Web 使用挖掘中,如何准确、及时、全面地采集用户使用数据是 Web 使用挖掘的重要前提和基础。文中分析了传统的基于 Web 日志进行 Web 使用挖掘

所面临的问题,对建立在用户浏览行为基础上的客户端数据采集的 Java Applet 实现技术进行了详细讨论。不同的数据采集方式有不同的特点。在实际工程应用中可根据自己项目的需求采取相应的数据采集方式。对于我校的 C++ 程序设计学习网站的个性化和智能化服务,分别从客户端、应用服务器端和 Web 服务器端进行 Web 使用挖掘的数据采集,在客户端主要采集学生的浏览页面和访问时间,它用 Java Applet 来实现,在应用服务器端主要采集学生注册的相关信息,然后结合 Web 服务器端自动记录的 Web 使用日志,如图 3 所示。

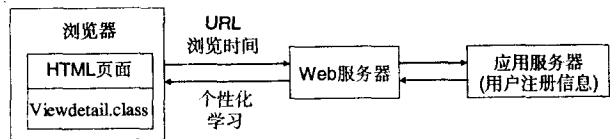


图3 个性化学习网站的学生浏览数据采集

#### 参考文献:

- [1] 涂承胜, 陆玉昌. Web 使用挖掘技术研究[J]. 小型微型计算机系统, 2004, 25(7): 1177-1184.
- [2] 向坚持, 刘相滨, 徐选华. 基于用户行为的 Web 使用挖掘数据采集技术研究[J]. 计算机与现代化, 2007, 12: 59-62.
- [3] Chen M S, Park J S, Yu P S. Data Mining for Path Traversal Patterns in a Web Environment[C]// In: Proceedings of the 16th International Conference on Distributed Computing Systems. Hong Kong: [s. n.], 1996: 385-392.
- [4] Yan Tak, Jacobsen M, Garcia Molina H, et al. From User Access Patterns to Dynamic Hyper text Linking[C]// In: Proceedings of the 5th International World Wide Web Conference. Paris, France: [s. n.], 1996: 1007-1014.
- [5] Borges J, Levene M. Data Mining of User Navigation Patterns[C]// In: Proceedings of the WEBKDD'99 Workshop on Web Usage Analysis and User Profiling. San Diego, CA, USA: [s. n.], 1999: 31-39.
- [6] Seiger M, Madsen M R, Langston J, et al. 点击流数据仓库[M]. 陆昌辉, 张光剑, 陈佐, 张丽, 译. 北京: 电子工业出版社, 2004.
- [7] 朱志国, 邓贵仕. Web 使用挖掘技术的分析与研究[J]. 计算机应用研究, 2008, 25(1): 29-32.
- [8] 刘立军, 周军, 梅红岩. Web 使用挖掘的数据预处理[J]. 计算机科学, 2007, 34(5): 200-201.
- [9] Schildt H. Java 2 参考大全[M]. 第5版. 周志彬, 吕建宁, 章小莉译. 北京: 电子工业出版社, 2004.
- [10] 王迪华, 刘臣勇, 刘立鹏, 等. JSP/Servlet——基于 Java 的最新网站建设工具[M]. 北京: 清华大学出版社, 2001.