Paper 154-27

Mining Web Server Logs: Tracking Users and Building Sessions

Don Koch, SAS Institute Inc., Cary, NC
John Brocklebank, SAS Institute Inc., Cary, NC
Tom Grant, SAS Institute Inc., Cary, NC
Richard Roach, SAS Institute Inc., Cary, NC

ABSTRACT

Much has been written about the limited usefulness of web server logs for mining of transactional and click-stream data. The truth is that web server logs are extremely useful for mining Electronic Commerce (e-commerce) as well as non e-commerce web sites. Server logs, when combined with minor, non-intrusive, site instrumentation can provide tremendous insights into how effectively your electronic channel is meeting its business needs. User tracking and session building are fundamental to all web-mining activities. Although using web server logs for this purpose can be a challenge it is not impossible. This paper will discuss how to track user activity and build sessions from web server logs. We will focus on data preparation and will not discuss web-mining techniques. Audience should be familiar with web technology.

INTRODUCTION

Web mining is a technology that can be used to help improve the business supported by a web site. Among the goals of web mining are to understand the customer's behavior and experience, determine which web designs, cross-sells and campaigns actually work and to improve the on-line experience of customers. An on-line business has an advantage over a real-world business in that collecting data on the activities of your customers is much simpler, can be automated and is much more complete. However there are several problems associated with getting an accurate accounting of customer activity, especially if you are relying on web logs. If you have the luxury of starting from scratch, you don't need to rely on web logs at all. Instead engineers would design web applications that more accurately record user sessions. For many on-line businesses, a complete re-engineering of their web site may be too costly, impractical or may not be among their short-term strategies. Even those business that are in the process of re-tooling their web site may still need to rely on web logs until their new strategy is fully in place.

Mining weblogs is and will remain a necessity for many on-line businesses for the foreseeable future. The purpose of this paper is to describe the problems with and solutions for re-building sessions from web logs so that web data may be used fruitfully for web mining.

TRACKING USERS

The ability to track user activity in a web log is crucial to building a mine-able warehouse. It's important to understand the difference between identifying and tracking users. Identifying users is the ability to link an on-line identity with a user's offline identity. Tracking users is the ability to determine the set of web log records generated from a single browser. Tracked users can remain anonymous; identified users do not. In order to identify users, a web log must contain personal information such as name or social security number or contain a unique identifier that can be joined with a database, which contains personal information.

For identifying information to appear in a web log, the user must supply the information either with a secure login or by filling in a form. Few if any successful e-commerce sites require their users to authenticate just to shop. Web applications, such as a checkout process, or an online form almost never log personal or sensitive information. When a user submits a form that contains personal information, the HTTP POST method is usually used instead of the GET method thus keeping the information private.

On the other hand it is very common for a web site to track its users. Tracking is essential for web mining since it is a key component in session re-assembly (grouping log records by user). However even the best user-tracking scheme suffers from the "de-heading" problem, which will be discussed later in this paper.

USER TRACKING TECHNIQUES

There are basically two ways to track users: ***cookies*** and ***URL re-writing***. Cookies can be further broken down into two broad classifications: ***server cookies*** and ***application cookies***. Because of their importance in preparing web data for mining activities the next few sections will discuss these techniques in some detail.

SERVER COOKIES

The web server software checks for and generates ***server cookies*** for each request received. The server checks for its cookie in the *"Cookie"* header of the request. If the request does not already have a server cookie then the server will generate a unique value for the user and send it back in the response in a "*Set-Cookie"* header. Upon receiving the response the cookie may or may not be "remembered" by the browser depending on whether the users has cookies enabled or not. The web server only records the cookies that the browser sends in the "*Cookie"* header when making a request. It does not record the cookie that it generates when the user first visits the site since that first request does not have a server cookie. This distinction is important, as it is the root of the de-heading problem.

Server cookies are very easy to implement and are completely transparent to the web application. Even a web site that is composed only of static content can be instrumented with server cookies with no change to the site content itself. The two most popular web servers in the industry support server cookies.

EXAMPLE IIS

For Microsoft Internet Information Services (IIS) you can install the Site Server User Identification Filter (an ISAPI filter), called **mss_log.dll**, which comes with Microsoft Site Server. This filter generates a cookie called "SITESERVER=ID" with a 32-byte GUID (globally unique identifier) which is guaranteed to be unique. An example of the IIS cookie as it would appear in a web log is "SITESERVER=ID=84aae92860f0917fc0ab3785ec1d37c7". Alternatively, if you are using ASP generated pages on your web site then you can take advantage of the ASPSESSIONID in the same way.

EXAMPLE APACHE

The Apache web server supports server cookies with the **mod_usertrack** extension. This module works much the same as mss_log.dll, setting a cookie called "Apache" for request that doesn't already have one. An example of the Apache cookie as it would appear in a web log is "10.40.12.61.67935594959155468832".

Although the two cookies, SITESERVERID and Apache, are generated differently, you can consider these cookie values to be randomly generated character strings that are guaranteed to be unique.

APPLICATION COOKIES

Web applications can also generate cookies in order to track users. Quite often a user will be directed to a page that checks for the existence of the application cookie and whether cookies are turned on or not. Uses of application cookies are as varied as web applications themselves. Besides "remembering" user state and preferences, they can also be used for user tracking.

URL REWRITING

Another technique used for tracking users is to re-write each URL on each page with a special tracking id in the query string of the URL. This technique only works with dynamic pages that are built on the fly with server-side technology such as Sun Microsystems Java Servlet API (servlets and JSP's), or Microsoft Active Server Pages (ASP). The web application container handles the creation of a user-tracking id so that applications don't have to.

You will find the user-tracking id that is created by URL re-writing not among the cookies in a weblog, but as one of the query string parameters. For example, when I browse a large on-line book retailer my pages are built with links that look like this:

```
<a
href="http://shop.barnesandnoble.com/gc/gc_buyno
w.asp?userid=174UFK2UYB">"
```

In a weblog this would look something like (I'm guessing here):

```
2002-01-14 16:19:41 123.456.789.123 GET
/gc/gc_buynow.asp userid=174UFK2UYB 200
```

The 6'Th field in this record is the URL query string. My userid is 174UFK2UYB. This id is used to keep track of my session. It is also stored in a cookie so that I can be kept track of across sessions.

SESSIONS

A session is the sequence of pages viewed and actions taken by a single user during a defined period of time. Quite often a period of inactivity, perhaps 30 minutes, can be a signal that the user's session has ended. Sites that require authentication may also provide users with the ability to "logout", thus ending a session. Other than logout or timeout there is no way to determine when a user terminates a session by looking at web log data. The "real" session is virtually impossible to reconstruct unless you can tap into the user's browser and record every click. But it's not always necessary to record the "real" session. It may only be necessary to reconstruct import milestones during the session, ignoring activities that don't relate to the core business of the site.

SESSION IDENTITY

To reconstruct each user's session from web log records, each record must have some piece of identifying information. Ideally the site is using some sort of user tracking as was describe previously. There are other pieces of information on each web log record that may seem like good candidates for session identity. Table 1 lists some of these and their attributes.

Table 1: Pieces of information in a typical web log.

Field	Relationship	Presence	Lifetime
IP Address	Many to Many	Always	None
User Agent	One to Many	Usually	Browser
Time Stamp	Many to Many	Always	None
Cookie	One to One	Second Hit	Persistent
Query String	One to One	Second Hit	Session

Relationship

- Many to Many - A specific value of the field can be used for more than one browser. A browser has more than one value in the web log.
- One to Many - A specific value of the field can be used for more than one browser. A single browser has only one value.
- One to One - A specific value of the field is only used for one browser. A single browser will have only one value.

Presence

- Always - There is always a value for this field.
- Usually - There is usually a value although it may be blank.
- Second Hit - This value is blank the first time a browser visits a site or the first hit of a session if persistent cookies are not used to remember the value across sessions.

Lifetime

- Session - This value is maintained throughout a session.
- Persistent - This value is maintained across sessions but may expire.
- Browser - This value is maintained until the browser is upgraded or changed in some way.

Categorizing web log record fields in this way helps to determine which field or combination of fields will be useful for re-building sessions. None of these fields are adequate by themselves. The IP address is not unique to a browser and indeed a browser may appear to have several IP addresses in a single session. This is because the vast majority of browsers sit on networks behind corporate firewalls or Internet service provider proxy servers. The web server sees the IP address of the proxy and not the browser.

User Agent (a browser version string) typically does not change unless the browser software is upgraded or changed in some way. The User Agent is however, not unique to a single browser. A high traffic web server may see thousands of different User Agents per day.

There are two problems with trying to identify a browser using server cookies. First a small percentage of the browsing population will not accept cookies from any site. We typically see anywhere from 3-7% of the records in a web log do not contain cookies. Users may elect to turn cookies off but in doing so they may then have to accept a lower level of service. The second problem is one we call "de-heading" which is described in the next section.

URL rewriting can solve the problem of users turning their cookies off since this method of user tracking does not rely on cookies. When URL re-writing is used in conjunction with cookies, it is possible to track users across multiple sessions. URL rewriting also suffers from the "de-heading" problem.

DEFAULT SESSION IDENTITY

In the absence of a cookie or query string parameter for user tracking, a concatenation of IP address and User Agent is often used for session identifier. Table 2 shows some summary statistics for a single day web log. Using IP or IP with User Agent doesn't resolve sessions very well. The best estimate of sessions is using Cookie Ids and re-heading (discussed below).

Table 2: Summary of web log covering 24 hours.

Total Page Views	92,000
Unique Cookie Ids	6,842
Unique Visitor IDs after re-heading	4,285
Unique IP addresses	4,548
Unique User Agents	950
Unique User Agent + IP Address	5,788

DE-HEADING

De-heading is the term we use to describe the miss-identification of the first request a browser makes upon visiting a site for the first time. Assume a site is using cookies for user tracking. Now

consider the very first visit a browser makes to the site. This first request does not contain the user-tracking cookie and so it is not logged in the web log. The first response sent back to the browser would set the user-tracking cookie. All subsequent requests made by the user will carry with them the user-tracking cookie.

For example a browser visits a site for the first time and makes 3 requests. Each request logs date, time, IP address, method, document requested and cookies.

```
2001-12-25 08:23:34 1.2.3.4 GET / -
2001-12-25 08:25:12 1.2.3.4 GET /p1.html UID=a
2001-12-25 08:32:23 1.2.3.5 GET /p2.html UID=a
```

In this example the first request to the site did not have any cookies so the cookie field is a single dash. Subsequent requests did have the UID cookie. If we rely on the UID cookie to rebuild this session then clearly it will be missing its head. This set of requests results in re-building two sessions; one with only one request the other with two. Also note that IP address is not a good choice to re-build the session since the IP address of this browser seems to change during the session (the proxy effect). These IP addresses are also not unique to this browser. As long as the user's browser is configured to accept cookies, then de-heading will only occur when the user first visits the site. Users that configure their browser to reject cookies from the site will have every session de-headed. Users that periodically delete their cookies will also appear as new visitors.

The same problem occurs when URL re-writing is used. If cookies are not used in conjunction with URL re-writing to persist the tracking id between sessions or if a user has cookies turned off then de-heading will occur for each session.

THE IMPACT OF DE-HEADING

There are two major results of a session loosing its head; loss of the session referrer and over counting sessions. Since much of web mining depends on accurate sessions, de-heading can have a significant impact on analysis.

The **session referrer** is the referrer of the first request of a session. It is important because it tells you how visitors arrived at your site. All other requests within a session will have referrers from within your site. When the session referrer is lost it appears that the user came to the site from within the site. Internal session referrers are legitimate if you are using an inactivity time out (typically 30 minutes) to define session boundaries.

For example consider this session that was referred to a fictitious site **me.com** from **www.yahoo.com**. This time each request logs date, time, method, document, cookies and referrer.

```
2001-12-25 08:23:34 GET / - www.yahoo.com
2001-12-25 08:25:12 GET /p1.html UID=a me.com
2001-12-25 08:32:23 GET /p2.html UID=a me.com
```

De-heading causes this session to be broken into two sessions. The first session, a single-hit session, was referred by Yahoo but appears not to have done anything interesting. The second session appears have been referred by **me.com**. The user appears to have started a new session from within the site itself.

What if the owner's of **me.com** had just launched an ad campaign that includes banners on affiliate sites and several popular search engine sites such as Yahoo. In order to determine their return on investment (ROI) for this campaign they need to know if the sessions referred by these sites actually made any purchases. De-headed sessions make that very difficult.

The second problem de-heading cause is to inflate the number of sessions. Session inflation results in incorrect conversion rates, ROI and other metrics that use number of sessions as a denominator. A site may spend $100,000 on an add campaign that appears to result in 4,000 new visits when in reality there were only 2,500 new visitors thus underestimating cost of acquisition by over 60%.

Table 3 shows the effect re-heading has on data from an actual on-line retailer. This site typically sees about 100,000 page views per day. The data from one day's traffic was processed with and without re-heading. Of the 358 de-headed sessions, 268 were re-united with their correct session referrers. The remaining 90 internal referrer sessions comprise two groups: sessions that were re-started after an inactivity timeout and thus legitimately had an internal referrer and sessions from users that had cookies turned off. The total number of sessions is reduced since one-hit sessions are rejoined with their proper sessions.

Table 3: Session counts (not all data shown)

	De-headed	Re-headed
Sessions with internal referrers	358	90
Sessions with no referrer	4,477	2,971
Total number of sessions	8,272	5,228

RE-HEADING

The solution to de-heading is re-heading. Unfortunately there is no magic bullet to re-heading. The first step is to understand how your web site works. Here are some strategies that we have found useful.

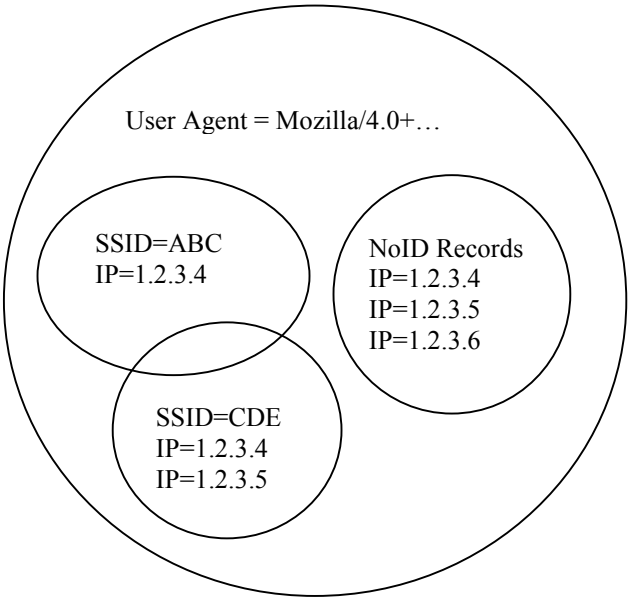
USER AGENT - DIVIDE AND CONQUER

The User Agent that a browser sends to the server is a browser version string. This value is virtually constant within a session. Although there is only a hand full of browsers being used today, we typically see tens of thousands of unique User Agents every day. The User Agent can contain a surprising amount of information about the browser and operating system of the client. By sub-setting the web log records first by User Agent we can make some simplifying assumptions about the no id (NoID) records; records that do not have the user tracking cookie. We also are able to handle large volumes of data since we can partition the data prior to processing. The figure below illustrates a set of records that have been divided by User Agent and by fictitious SITESERVER ID (SSID). This population of records falls neatly into those with an SSID and those without an SSID referred to as NoID records. The overlap of the two groups with valid SSIDs shows that some records in these groups share some attributes in common such as IP address. The goal of re-heading is to move records from the NoID group into one of the other groups. This example was constructed to illustrate the problems with using IP address to do this. Which group should you assign the NoID record with an IP of 1.2.3.4?

USING TIME WISELY

You may be able to use time to determine which session a NoID record belongs to using the record time stamp. For example, if you know that the "index.html" page of your site immediately redirects the visitor to another page, perhaps to check for enabled cookies, then you can use the fact that a session's first request will occur within a few seconds of the next request. On the other hand any first request that is more than a certain interval away from the rest of the session, say 30 minutes, can be ruled out. Time stamps can also be use in conjunction with IP address to resolve ambiguity as a tiebreaker; the head being assigned to the session that it is close to in time.

A further refinement can be made if we assume that there is only one head per session. In the case of an ambiguity, the head can be assigned to the session that still remains headless. We may also rule out certain sequences if we know a user cannot navigate a particular path.



IMPLIMENTATION

Any algorithm that implements re-heading must be flexible enough to handle the peculiarities of the site. The process is also an iterative one, requiring multiple passes through the data. Performance and memory constraints can become an issue with large web sites. Partitioning by one or more fields can help increase performance. SAS® WebHound™ software is planning to support re-heading in a version to be released later this year.

CONCLUSION

Web server logs present a unique challenge for web mining. They are however a ubiquitous, cheap and somewhat standard source of data to mine your e-commerce channel. With a good understanding of how your site works it is possible to properly prepare your web logs for data mining. At a minimum server-generated user tracking cookies should be employed. If a web application environment is employed then URL rewriting can also be used to track users. These changes are very unobtrusive to a site and can reap large benefits in analysis. Further re-engineering of web site and web applications can provide more accurate data. The principles laid out in this paper can be used in any re-engineering effort aimed at better understanding of your on-line customers.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Don Koch
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27502
Work Phone: 919-531-4823
Fax: 919-531-4444
Email: don.koch@sas.com
Web: www.sas.com