

Web-Intelligence: A Primer

Don Henderson, PricewaterhouseCoopers LLP
Ralph Mittl, PricewaterhouseCoopers LLP

Abstract

Web-Intelligence (also referred to as e-Intelligence) is the application of business intelligence and analytics to data resulting from activity at a web site. Given the advent of e-commerce, such data is becoming very important to organizations as they strive to:

- ❑ Understand who their customers and prospective customers are
- ❑ Provide appropriate content to their web site customers
- ❑ Leverage other information they have about customers and prospects from other channels

Much attention is given to concepts like web-site personalization engines, click-stream analysis, predictive modeling, web-site optimization (the list goes on).

In order to fully leverage this data, it is essential to have an understanding of what data are available and the types of questions that data can help answer. The next step is then to use a proper data warehouse platform to support the access, management and analysis of that data.

This paper will provide an overview of the key concepts involved in making effective use of web log data.

Basic Definitions

There are a number of metrics and terms that are discussed in the web-intelligence space. In order to leverage the data available from a web site, it is essential to understand exactly what the metrics are, and their corresponding limitations.

One of the most commonly discussed metrics is **Hits**. When a user visits a web site, each request is logged, and each of those requests is counted as a hit. The Hits metric is of interest to those who manage web sites and are interested in load, demand, etc. However, Hits is a metric whose value can be misleading. For example, if a single web page has 15 graphics, then whenever a user makes a request to see that page, there are 16 hits. One hit is counted for the main web page, and one for each of the 15 graphics.

The **Page View** is an attempt to deal with this shortcoming. A page view counts the number of requests to load a single web page, regardless of the content on the page. This metric is often measured by excluding from the counting any graphic or multimedia files. However, there are a number of other confounding factors that impact page views. For example, consider an HTML page that is a frameset, with two embedded frames (or panels). The question is whether to measure **Page Views** for the individual panels (which are also HTML pages) or the frameset definition page. Thus, this could be considered a single **Page View** for the frameset or single Page Views for each frame (or pane).

The number of **Visits** or **Visitors** is a measure of how many times any user has viewed any page on a web site within some prescribed time period (a **Session**). Note that if a user leaves the site and returns after the prescribed time period ends, this is interpreted as two visits.

Summarizing these, consider the scenario where a single person visits a site and views 7 pages (no frames to keep the

example simple), each of which has 5 embedded graphics. The values for these metrics would be:

- ❑ **Hits:** 42
- ❑ **Page Views:** 7
- ❑ **Visitors:** 1

Next, let us consider the concept of a **Unique Visitor**. A unique visitor is an identifiable individual who visits a web site and is typically a major interest to most business users. However, quite often, the term **Visitor** is (incorrectly) used interchangeably with **Unique Visitor**. Measuring unique visitors is problematic because, at its core, the web is an anonymous/stateless environment. Thus, any attempt to measure or quantify unique visitors requires the application of business rules.

The term **Click-stream** (or click-stream data) is often used and is also subject to many interpretations. Some uses of the term click-stream data refer to any/all the data collected from the activity of users/visitors requesting pages from a web site. In the context of its intended use, however, the term click-stream is intended to mean a data-based representation of the path or sequence of activity (e.g., pages viewed) while a user navigates through a web site. It is this pathing data that is typically of high interest (e.g., what click-stream paths through my web site resulted in more or higher-value purchases?) to most business users. It is also this type of data that is most desirable for Data Mining.

Cookies are widely used, reviled and misunderstood. A cookie is simply a text file that is stored on a user's local machine and contains data specified by a web-server. The data is specific to that server and the data stored in a cookie are only available to the server that provided them. A standard Set-Cookie script allows the web server to specify the following:

- ❑ **Name**
An arbitrary string containing the name of the cookie.
- ❑ **Value**
Any specified value, as a string, to be stored in the cookie.
- ❑ **Expires**
The datetime the cookie should expire (Greenwich Mean Time). Note that a value can be provided that indicates that the cookie should never expire.
- ❑ **Domain**
Domain name of the server(s) that can read the cookie. A domain cannot read a cookie unless the server that created the cookie grants permission explicitly.
- ❑ **Path**
Pathname in the domain for which the cookie is valid.

The form and location of these cookies depends on the browser being used.

Cookies can be used to provide a variety of facilities, including:

- ❑ The storage of user preferences
- ❑ A unique user identification
- ❑ Market Basket data

While cookies themselves are somewhat innocuous, they have been used in the past to do things that many users feel to be inappropriate or invasive. Thus, the use of cookies must be carefully evaluated in any web environment.

A cookie is often used to identify you when you return to a web site (e.g. AMAZON.COM). To accomplish this, the cookie stored some identification key that the site could translate to your name or some other value you have provided to that site in the past. When you visit the site again, the data in the cookie is provided by

your browser to the web server and that information is then used to populate the page you see.

Web Log Data

Web servers log their data to log files, creating the most common source of data. There are a number of standards, the first being **Common Log Format** (or CLF). The successors, by and large, are the **Combined Log Format** or the **Extended Log Format** (ELF) which includes additional commonly used/needed fields.

The ELF Data

The ELF fields include, among others:

❑ Host

Host is the Internet address of the browser or other agent making the HTTP request and the location where the response will be sent. The value for host is the numeric IP address (e.g., "124.11.121.11") and that is the value that will be typically seen in web logs. Most web servers can resolve this address into a text domain using an Internet query protocol called reverse DNS (**D**omain **N**ame **S**erver) lookup. This is a process whereby the IP is replaced by the domain name (e.g., www.microsoft.com). This makes logs more readable, but can increase the load on the server dramatically if done in real time. More commonly, if DNS lookup is needed, the IP lookup is integrated with the Data Warehouse ETL process.

At first glance the results of DNS appear to be potentially very useful (e.g. what percent of my visitor are from SAS.COM). However, upon examining the data it is only useful for a limited set of reporting functions. For example, a **B2C** site will typically find that the vast majority (e.g., upwards of 80%) of the IPs resolve to AOL.COM. Thus, knowing the actual domain name

provides little discriminatory value.

As discussed later, this IP value is commonly used to identify visitors and sessions. The IP value remains constant during a browser session, and can be used to tie events together where a more reliable mechanism such as a cookie or server-generated session ID is not available.

❑ Ident

The ident data element is an arbitrary identifier that can be supplied by client applications that support the identd (identity daemon) protocol. Most common browsers do not provide this value and so this field is rarely if ever used.

❑ Authuser

Authuser is a user ID that the web server will prompt the user for if HTTP authentication has been enabled on the web server. The user must enter a valid user id and password before the web server will provide any pages to the user browser. Only the user id is stored in the web log as providing the password would be a security breach.

Figure 1 shows an example of a prompt that the browser will display if the web server signals that HTTP authentication is required.

Figure 1. HTTP Authentication



Many people incorrectly associate the

Authuser field with requests that are made to a secure server using the HTTP secure sockets layer (SSL). Such URLs begin with HTTPS instead of HTTP and are commonly used at sites where sensitive data (e.g., credit card numbers) are entered. The Authuser value is logged whenever HTTP authentication is enabled and that value is written to every web log record even though the user is only prompted once.

❑ **Time**

Time is usually the time when the web server completed the response to the HTTP request. It is usually is set to GMT (Greenwich Mean Time). For web sites that are supported by multiple servers, atomic clock should utilities be used for all servers to ensure synchronicity.

❑ **Request**

The request field contains the actual request line from the browser, for example "Get /mypage.html HTTP/1.0".

In this example, GET is the HTTP method, the next section is the Uniform Resource Locator (URL), and HTTP/1.0 is the protocol version being requested by the client. The two most common request methods are GET, which requests an object from the server, and POST, which sends information from the browser to the web server.

❑ **Status**

Status is the three-digit status code returned to the browser from the server and indicates whether the page was returned to the user's browser (and, if not, why not). Example values include:

- ❑ 200 (OK)
- ❑ 302 (Moved Temporarily)
- ❑ 404 (Not found)

❑ **Bytes**

Bytes is the count of bytes returned to the client by the server. It is seldom used, but is important to certain sites. For example, a site providing video or audio files which are large would likely make more use of this field (e.g., how many of my first-time visitors are downloading files larger than 300K).

❑ **Referrer**

Referrer is a text string that contains the referring page, i.e., the URL of the page that contained the link a user clicked on to get to the current page. The referrer field allows you to trace how a user got to a page by navigating backwards through the web log.

If a page contains images or embedded pages (e.g., framesets or iframes), then the referrer value for the page component is the container page that includes the references.

It is the referrer field that typically provides the data source for **click-stream data**.

❑ **User-agent**

The user-agent is the name and version of the client software making the request, and the corresponding operating system. This can be used to ensure that only content that can be supported is sent to the client. Unfortunately, this is very difficult to accomplish, as there is still not a widely used industry standard. For example, both Internet Explorer and Netscape have their own proprietary version of Dynamic HTML, Cascading Style Sheets, etc. and their own extensions as well as supporting some but not all of the supposed industry standard data elements.

On an Intranet (as opposed to Internet) site, this is typically a more manageable

problem as standards can dictate what the supported browser (and version) is.

In addition to browsers, the user-agent field can also contain values for search engines, spiders, or web-bots that crawl the Internet (or an Intranet) to find, index and catalogue text.

An application that makes an HTTP request can provide any value it wants for the user-agent field. This field can also be easily changed by the end-user in their registry or other appropriate system location. In developing their WebHound solution, SAS Institute identified well over 1000 different values for user-agent based on just examining the web logs for their external site.

Data Usage Problems

When using data to investigate an issue, it is important to understand the nature of the data, how it was collected, and how it can be used. There are a number of issues relating to the quality of web intelligence data that must be considered.

The Visitor Problem

At the core of any web-intelligence effort is the need to uniquely identify visitors. However, as described above, the web is an anonymous and stateless environment. Thus, the identification and definition of a unique user must be dealt within the design of the web site. For some environments, uniquely identifying a user is straightforward, while for others it can be next to impossible.

For an Intranet site that requires authentication to the web-server, the user's id is automatically captured by the web-server, is written to the standard logs, and is easily made available to any web application server.

For the typical Internet site, web server authentication is not an option. Thus, other techniques must be used to identify visitors.

Cookies are commonly used to identify unique users. When a user visits a web site, the web server can request that a cookie be created on the user's pc and store some anonymous but unique identifier in that cookie. Any future visit to that web site will include the cookie value and so it is possible to identify a unique user (though this technique does not provide any other information about who the user is). There are a number of problems with this option:

- ❑ Some users may disable cookies
- ❑ If multiple users share a single machine (e.g., a home PC), the unique visitor identification is really identifying a household and not an individual
- ❑ A user may roam (e.g., use more than one pc) and have different cookie values on different PCs.
- ❑ The cookie on a specific PC represents that PC and not the specific individual user.

Regardless, the use of cookies is a widely used technique that can suffice for many situations.

Another commonly used technique is to request that the user log in or identify themselves to the site. However, this requires that the web site provide some incentive for a frequent visitor to log in vs. browsing *anonymously*. It also requires that the web site be designed so the values are propagated as the user navigates through the web site.

The most widely used technique groups together log records with the same IP number and assumes that they represent a single visitor if the time between successive

requests is less than some value (30 minutes is typically used).

The use of the IP number to identify unique visitors has limitations. First and foremost, the IP is not tied or associated with a specific visitor. A few scenarios where multiple users could share a single IP include:

❑ **Multi-user Machines**

More than one person is using the same machine to browse the web site, and since the IP is associated with the machine, multiple users are indistinguishable.

❑ **Proxy Servers**

A proxy server is used by many organizations and ISPs to cache commonly requested content. When a user makes a request for a page, it is the proxy server that is making the connection to the web site and so the IP of the proxy server is logged. Thus all of the individuals going through that proxy server are indistinguishable.

❑ **IP Reassignment**

If a user logs off (e.g., disconnects from their AOL account), his or her IP is now available for use by someone just logging on.

It is also possible for a single user to have two IPs. Many Internet Service Providers (ISPs) have a policy that if there is no activity for some time period (e.g., 15 minutes), the user's connection is terminated. Consider the scenario where a user is browsing and stops on a page, gets distracted, and returns 20 minutes later and continues to browse. When they click on a link, a new ISP session is begun and they likely have a new IP. Alternatively, for sites with multiple proxy servers, any given request from a client to an external site can appear to come from any one of these proxies.

The Click-stream Problem

The *holy grail* of web intelligence is the use of click-streams or pathing data in order to gain insight into customer behavior. For example, if you can identify patterns in the click-stream that result in a higher propensity to buy, or that increases cross selling of related products, the usability and profitability of a web site can be improved.

There are (at least) two stumbling blocks to making use of click-stream data. First is ensuring that the click-stream path is collected completely and correctly. A second issue is the size of the data volume.

Consider the problem of ensuring that the click-stream path is collected completely and correctly. The data that is available is based on activity/requests made to the web server. However, there are external factors that can prevent a complete click-stream (from the user's perspective) from being collected. The issue is caching of pages – either by the user's browser or by a proxy server.

Follow the click-stream path below:

1. user starts at page New Orleans
2. clicks on a link to Shopping
3. clicks on a link to Clothing
4. clicks on a link to Jax Brewery
5. browser back button, to return to New Orleans
6. clicks on the link to Shopping again
7. goes back to New Orleans
8. clicks on a link to Mardi Gras
9. clicks on a link to Mardi Gras America
10. decides to buy
11. and continues.

So the user's actual click-stream path is:

**New Orleans:Shopping:Clothing:Jax
Brewery:New Orleans:Shopping:New
Orleans:Mardi Gras:Mardi Gras America,
etc. . .**

Due to the fact that the browser has cached the pages, when the visitor uses their back button, there likely is no record at the web server of that activity, so the data that is logged might be:

**New Orleans:Shopping:Clothing:Jax
Brewery:Mardi Gras:Mardi Gras
America, etc. . .**

Further, suppose that the pages **New Orleans** and **Shopping** (being the main pages at a popular site) were cached by a proxy server (and lets ignore the issue of identifying the visitor for this example). The resulting click-stream recorded at our web site might be:

**Clothing:Jax Brewery:Mardi Gras:Mardi
Gras America, etc. . .**

An analysis of this data might yield the conclusion that this is **THE** path that leads to higher-value purchases when in fact, pages **Clothing** and **Jax Brewery** are digressions and, in fact, many visitors to Clothing and Jax Brewery may actually abandon the site. The important path might be **New Orleans:Shopping:Mardi Gras:Mardi Gras America**. However, that is not what is recorded in our click-stream.

A compounding factor is browser favorites. By saving an intermediate page in the click-stream path as a favorite and returning to it later, the actual path that the user took may be spread out over several sessions. As a result, when the user makes the decision to buy, it is a truncated click-stream that is recorded in the log.

Next, consider the issue of data volumes. The click-stream paths that can be

recorded are both very long and very voluminous. The patterns of interest are likely a small subset of a much larger click-stream. If one considers the entire click-stream, then many of the paths are going to be unique. The cardinality of the data is very high and so any analysis of the data is difficult at best.

As yet, no one has identified an apposite solution to these problems. However, the problem can be made tractable by scaling back what is examined. Instead of examining all of the click-streams, a site should consider identifying a small subset of pages of interest, and then build the click-stream paths for just that subset. There are at least two techniques that should be considered:

1. **Pathing To**

Identify an ending page of interest (e.g., checkout) and then examine the click-streams that lead to this page.

2. **Pathing From**

Identify a starting page and examine the click-streams that start from this page.

The Web-Intelligence Platform

In order to provide a robust solution that integrates the data/information available from web logs, it is both necessary and appropriate to consider the design of both the web site as well as the data warehouse. Each of these will be covered briefly in the following subsections.

Web Site Design

Depending upon the scope and purpose of the web site, there are a vast array of techniques and methodologies that can be employed. While many are specific to the scope/purpose, some are more general in nature. Some of the common design considerations are included below.

❑ **Dynamic vs. Static Pages**

Web sites can be composed of a combination of both dynamic and statically generated pages. The primary advantage of static web pages is that such pages can be served quickly and easily. Dynamic pages can be customized to the user, and tracking access to them is less likely to be impacted by caching issues (by either a proxy server or the user's local browser cache). Dynamic pages place a higher demand on the web server. Sites can be built using either of these techniques or a combination. There are a number of techniques that can be employed for dynamic page generation, including:

- ❑ Common Gateway Interface (CGI)
- ❑ Servlets
- ❑ URL rewriting

❑ **Content Labels**

Content labels for pages allow the page events to be classified and coded for later analysis. These labels can either be coded manually, or may be generated by application files or directory structures automatically. Such an index can then be integrated with the web log extract system to be used as part of the Extract-Transform-Load (ETL) system. For both static HTML and dynamic HTML pages, tables must be set up and rigorously maintained. Possible classifications for content labels include:

- ❑ Page source (static, dynamic)
- ❑ Page template (catalogue, index)
- ❑ Page function (site index, product catalogue, FAQ, announcement)
- ❑ Item code (product ID)

And others as mandated by the business requirements.

❑ **Use of a Null Logging Server**

There are numerous names for this

technique. It has been made popular (and to some extent, infamous) by Doubleclick. The basic idea is to embed a reference on a page to an image (or a cgi reference, etc.) that is located on that server. By embedding such an HTML tag into a web page, a single web log record is written on the null logging server that contains the requested image as well as the page that requested it (in the referrer field). That data can then be directly used for **click-stream** analysis. One technique commonly used to do this is to use an HTML IMG tag to request a one pixel sized transparent .gif file. This results in a record being written to the server web log. By using a one-pixel .gif (also called a **webdot**), there is a minimal impact on the download time because of the small size of the picture. Thus, it can be done relatively transparent to the user.

Alternatively, instead of using a separate web server, the **webdot** can be served by the same web server. This requires more effort during the ETL process in order to identify the **click-stream**.

❑ **Unique URLs**

If the site is part of an overall CRM or Marketing Automation solution, then it is likely that prospects, customers, and users have received an email with a link to the site. Such URLs can be made unique to both the user receiving them as well as the marketing campaign. In order to fully leverage such data, it is necessary to make sure that the entire browsing session can be linked to the page. One way to do that is to count on the use of the referrer field to build the **click-stream** path. Alternatively, the site can be designed to propagate the values on the original URL (e.g., to identify both the user and the campaign).

❑ **Exit Point**

Tracking the last page of a site that the user visits is of very high interest and value. Unfortunately, it is not easy to identify. While the referring field will tell you what page the user navigated from, there is no way for the web log to contain the page for which the user left (though if the page is on the same site, it can be determined). For example, if your site contains links to external sites, when a user clicks on one of them, there is no information written to your server's web log recording that action by the user. Since this is useful and important information to have, many sites will employ what is called a redirect page. When the user clicks on such a link, they can be brought to a page (either an HTML page or a cgi, asp or jsp application) that indicates that they are leaving the site. Then the site either makes them click again or automatically transfers them after a short time. The record written to the web server log contains both a page on the current site (thus is it logged) as well as the site the user is being transferred to.

❑ **Entry Point**

Tracking the first page a visitor visits is also important. Fortunately, this can be determined in a straightforward manner once visitors/session tracking has been addressed. The entry point will typically be the first page visited at the site for any selected time window that has either a URL for another site in the referrer field (this is useful data as it identifies where the visitor is coming from) or no value at all for the referrer field. The referrer field can be blank for a number of reasons, the most notable causes are:

- ❑ The user directly keyed the URL in the browser address field, or

- ❑ A previously saved favorite was used, or
- ❑ The user received an email with the link, which they clicked on to access the site.

Warehouse Design

There are a number of vendors and products that can be used to create a warehouse to support analysis of web log data, such as:

- ❑ WebHound from SAS Institute
- ❑ eSite from Informatica
- ❑ WebTrends
- ❑ Accrue
- ❑ NetGenesis

They each have their strengths and weaknesses. Instead of trying to compare these offerings, this paper will focus on general design principles that should be factored into the decision for the Warehouse platform.

Any warehouse platform must be scalable and extensible and should include, at a minimum, the following features or capabilities:

❑ **Warehouse Centric**

The platform should be fully integrated with a data warehouse environment and should be based upon a data model that addresses both the specifics of web log data, and is extensible so that site specific characteristics and fields can be integrated into the warehouse.

❑ **A Standard Web Log Data Model**

The model should address the following constructs at a minimum:

❑ **Hosts**

A table that contains information about all the hosts (internal and

referring) that appear in the web log.

- ❑ **Pages/URLs**

Each unique page or URL that is surfaced by the site should be identified and should be integrated with the Content Label functionality as earlier described.

- ❑ **Sessions**

Each unique session should be identified and information about it should be available from a sessions table.

- ❑ **Visitors**

Each visitor should be identified in a table. If IP is used to identify visitors, then the visitors and the session tables will be one in the same. If some mechanism (e.g., cookies) is used to identify visitors then the visitors table should be linked to the sessions table using a one-to-many relationship.

- ❑ **Paths**

Each unique path should be stored in a table. It may also be appropriate to define a table of unique paths by session or visitor.

- ❑ **Integrated with Other Channels**

In order to maximize the effective use of data collected from web logs, the warehouse platform must enable easy integration with other enterprise data (e.g., customer history, product information, etc.). If the web log warehouse contains only information about web-site access, its usefulness will likely be limited to optimizing the web site itself rather than providing additional data that can be integrated into a CRM environment.

- ❑ **Static Reporting**

There must be a rich set of publishable reports that contain information about

web-site access and usage as well as information of business value (e.g., how many sessions, visitors, most common entry/exit points, typical paths to purchase, etc.)

- ❑ **OLAP Tools**

Since not all reports can be generated statically, seamless integration with OLAP tools to allow business and technical users to slice-and-dice the data for discovering trends and patterns is an essential component of any warehouse platform.

- ❑ **Data Mining**

Web logs can provide massive amounts of data and so integration with data mining tools that can be used to mine and discover information contained in the web logs is important. Data mining the web log information is an ongoing activity and should be targeted to address specific questions. There must be an ability to create data mining databases specific to certain problems. Trying to automatically mine the entire web log data is likely to be not nearly as productive as mining subsets of the data extracted for specific questions.

- ❑ **Web-based Platform**

Reports and access to the data should be web-based and deployed to any user with a browser (with appropriate security, as necessary). It is an oxymoron to require proprietary desktop applications to browse and mine web logdata.

Summary

This paper has provided a brief overview of the fundamental elements of web-intelligence. Questions for the authors can be directed to:

Don.Henderson@us.pwcglobal.com

Ralph.Mittl@us.pwcglobal.com