

Appendix A

COUNTER/JISC Research Project for Usage Filters: Methodology

Overview

The purpose of this research study is to introduce an enhanced filter and/or new metrics that will eliminate or at least dampen the effect a user interface may have on reporting of usage data. The proposal identified a new filter to eliminate unwanted HTML requests and a new metric for Unique Article requests in a session. This document will describe the methodology that will be used to define the attributes of the filters and to determine their reliability.

Unwanted HTML Filter

The goal of this filter is to compensate for the effect of interfaces where the HTML is displayed automatically with the detailed display as well as in cases where the user automatically is presented with the HTML version of an article when linking in from external platforms. This filter will detect and thus not count HTML full text requests that would not have otherwise been requested by the user. The Unwanted HTML Filter would be applied after or in conjunction with the double-click filter.

Because we have no way of logging a user's intention, we will attempt to find patterns in the usage logs that can be used to detect, with a reasonable level of reliability, when a user would not have otherwise requested the full text that was automatically displayed.

IMPORTANT NOTE: this filter would not need to be applied to interfaces where HTML is not automatically displayed when linking in or when browsing results at the detailed level.

The prerequisites for the transaction log for the test:

- At minimum, HTML requests, PDF Requests and Abstract requests must be logged

Transactions can be sorted by the elements that uniquely define a user session (e.g. user session cookie and next by Machine Cookie or IP (if MC is not available)) and within a sessions by time

- Transactions within the log for the user session have time-stamps with a granularity no greater than one second
- If multiple servers handle sessions, logs will be consolidated into a single transaction database for the study (all activity captured)
- If multiple servers handle activities for a single session, the clocks on the servers must be synchronized.

- If auto-display of HTML is a customer option, it must be possible to segregate usage into two groups (auto-HTML on and auto-HTML off).
- Select all sessions with a sessions length > X hours and store these in a separate file. This action is to remove data-mining robots or link-checking software that may be systematically downloading data.

Step 1: Preparing the usage data.

- Remove double-click activities from the transaction logs
- Pre-process usage data as necessary for test. For each transaction the following columns are wanted, at a minimum:
 - User Token /SessionID
 - Transaction sequence in the session
 - IP Address of user
 - User Machine Cookie
 - CustomerID
 - External Link-in Session (Y/N)
 - Auto-HTML on (Y/N) (may need to be derived from customer settings)
 - Type of request (Search, abstract request, full text request, etc.)
 - Format requested (HTML versus PDF)
 - UniqueID of Journal
 - UniqueID of Article (only for item requests)
 - Time-stamp of activity
 - Elapsed time until the NEXT user-initiated action in the session (to be calculated)

Step 2. Plot the Elapsed viewing time for wanted HTML

- Select only Full Text Request where format is HTML and Auto-HTML is off
- Analyze the Elapsed time field.
- Create a frequency table of customers per second of elapsed time.
- Use the frequency table to determine possible thresholds, where:
 - 100% of elapsed times exceed this value
 - 98% of elapsed times exceed this value
 - 95% of elapsed times exceed this value
 - 90% of elapsed times exceed this value
 - 80% of elapsed times exceed this value

Step 3. Test reliability of Step 2 results

- Perform step-2 tests separately for a number of customers (100?)
- Calculate deviations from the average Step-2 results
- For further confirmation, EBSCO and Elsevier can repeat this step for a number of common customers to ensure like results are coming from separate interfaces.

Step 4. Calculate average HTML requests per journal per customer for customers with autoHTML off

Step 5. Calculate average HTML requests per journal per customer for customers with autoHTML on

Step 6. Calculate average HTML requests per journal per customer for customers with autoHTML on and elapse time filter applied. Test for various threshold found in step 2

- 100%
- 98%
- 95%
- 90%
- 80%

Step 7. To check for over filtering, calculate average HTML requests per journal per customer for customers with autoHTML off and elapse time filter applied. Test for various threshold found in step 2

- 100%
- 98%
- 95%
- 90%
- 80%

Step 8 Calculate deviation between Customers with auto-HTML off (step 4) and those with Auto-HTML on with filters applied (step 6). Break out by threshold.

Step 9 Determine the degree of correction for customers with auto-HTML on by comparing results of step 5 and step 6. Breakout by threshold.

Step 10 Calculate the degree of error introduced for customers with auto-HTML off by comparing results of steps 4 and steps 8. Break out by threshold.

Step 11 Evaluate results and select time threshold from step 9 that most effectively corrects auto-HTML (bringing average in line with Step 4 averages)

Step 13 Using the threshold found in step 11, plot the deviation of customer-level results for at least 100 customers for step 4 results and step 6 results. The graph would show average HTML requests/customer/journal (one decimal place) on the X axis and number of customers with that value on the Y axis.

Step 14 Compare the pattern of the two lines on the chart and look for correlation.

Success will be when the threshold number found in Step 10 will dampen auto-HTML results sufficiently such that the average HTML requests per journal per customer is in line with the same number for customers with auto-HTML off, and the distribution of averages per customer is essentially the same as for our control group with auto-HTML display off.

Unique Article Request Count

This is a new metric that would be determined by examining the full text request transaction logs after the double-click and Unwanted HTML filters have been applied. As its name suggests, this metric is designed to remove the over counting of articles due to the interface effects in which the same article is retrieved in different formats (HTML, PDF, etc.) or for different output options (display, email, print).

Note: we will only propose this filter in case the Unwanted HTML filter will not produce sufficient results. In itself the Unique Article Request (UIR) is very much convincing but if we can get results by using the Unwanted HTML only, this would make a lot of sense. The UIR will remove valid and meaningful requests and keeping up our usage numbers is very much key.

Goals of the tests:

- Prove that obtaining a unique article count per session is possible
- Determine if session time-slicing is a viable option for reducing processing needs while retaining substantially the same results.
 - Determine the smallest time-slice that would still provide good results
- Determine if using logged IP address as a session identifier with a session time-slice approach.
 - Determine the smallest time-slice that would still provide good results

Prerequisites and assumptions

The same as for the previous series of tests.

Step 1: Preparing the usage data.

- Remove double-click activities from the transaction logs
- Pre-process usage data as necessary for test. For each transaction the following columns are wanted, at a minimum:
 - User Token /SessionID
 - Transaction sequence in the session
 - IP Address of user
 - User Machine Cookie
 - Session time-slice (a sequential number – this will be populated later)
 - CustomerID
 - Auto-HTML on (Y/N) (may need to be derived from customer settings)
 - Type of request (Search, abstract request, full text request, etc.)
 - Format requested (HTML versus PDF)

- UniqueID of Journal
- UniqueID of Article (only for item requests)
- Time-stamp of activity
- Elapsed time until the NEXT user-initiated action in the session

Step 2: Calculate the number of full text article request by customer in HTML format

Step 3: Calculate the number of full text article requests by customer in PDF format

Step 4: Using the sessionID, and the unique article ID, count the number of unique articles requested by customer.

Step 5: Report results of Step 2..4 and calculate averages per customer (this is our control group)

Step 6: Introduce a session time-slice of 5 minutes by reprocessing the usage file and inserting a sequential value in the Session Time Slice field using start time of the sessions and the time-stamps of each transaction.

Step 7: Using the sessionID and Sesssion Time-slice, and the unique article ID, count the number of unique articles requested by customer

Step 8: Report results of Step 2..4 and calculate averages per customer

Step 9: Evaluate results by comparing to the results of step 5 to calculate the deviation.

Step 10: Adjust the time-slice value and repeat steps 6-9 until sufficient data has been obtained to determine the session time-slice value that results in a deviation no greater than 2%.

Step 11: Use the user's IP address as the sessionID and repeat steps 6 through 10 until an optimal time-slice is obtained for IP addresses.

Success will be when we can show that a large volume of transactions can be processed without excessive load (subjective) on the data warehouse and that a time-slice can be applied to IP address-based session identifiers to provide comparable results to unique articles per session counts determined with user-level session identifiers.