Identifying User Actions from HTTP(S) Traffic

Georgios Rizothanasis[†], Niklas Carlsson[†], and Aniket Mahanti[‡]
[†] Linköping University, Sweden
[‡] University of Auckland, New Zealand

***Abstract*—When understanding modern web usage and providing optimized personalized service, it is important to identify the HTTP(S) requests directly caused by user actions like clicks and typing web addresses. With a majority of HTTP(S) requests being due to content that has not been explicitly requested by a user, the problem of identifying user actions at proxies or middleboxes becomes non-trivial. We present an automated evaluation framework for identifying user actions while also automatically providing a "ground truth" of the user actions. We utilize the framework to compare the performance of timing-based and HTTP-aware request classifiers, including timing-based classifiers operating on both per-request and per-connection basis to identify user actions. We emphasize the value of diverse information used by the classifiers when comparing identification accuracy both among classifiers and relative to the browser-based ground truth. Our classifiers can be useful to better understand users' web usage and connection prioritization.**

I. Introduction

Much effort has been devoted to use passive measurements to analyze and characterize Internet traffic [1], [8], [10] and the services providing web content [7] from a network operator's viewpoint. However, these works typically do not distinguish the content that the user originally requested (e.g., by entering a new URL into the browser or clicking a link) from all other objects downloaded in the background, but instead weigh all web requests the same. The results of these studies are therefore not representative of users' website usage.

Due to the complexity of the modern web [4], it is a non-trivial problem to identify the requests associated with actual *user actions* from among the large number of downloaded objects. For example, a single request for the Huffington Post website results in the download of 408 objects from 113 unique domains. A similar analysis by Butkiewicz et al. [4], of 1,700 popular websites showed that the median landing page consists of at least 40 objects, requested from 10 or more servers, most of which are operated by third-party services. In this work, we refer to objects not explicitly requested by users as *drag-along* content.

While researchers have proposed multiple techniques [1], [14], [17] to identify user actions from network data such as proxy traces, these studies are typically limited by two things. First, they focus only on HTTP traffic and do not consider HTTPS traffic. Due to the increasing use of HTTPS this may render many of these techniques useless in the future. Second, they typically either do not properly evaluate the accuracy of the model or require manual labeling of some training and/or evaluation data [13]. With large volumes of drag-along content (e.g., images, scripts, and ads) associated with the originally requested web page or other domains, manual labeling is time consuming, difficult, and error-prone.

This paper makes two contributions. First, we develop and evaluate a novel evaluation framework for generic request classifiers for user action identification. Here, we use the term "classifier" to refer to a set of rules used by the operator to automatically identify and label user actions. Our framework provides flexibility in applying several classifiers on the same traces (thus, allowing fair head-to-head comparisons of classifiers) as well as automated labeling of HTTP traces. It includes a browser instrumentation to label the true user actions (referred to as the *ground truth*), a Squid proxy that records all HTTP requests (referred to as the *proxy logs*), and a custom-built Selenium-based crawler (implemented as a browser extension) that can emulate different client click streams and user behaviors.

Second, we extend the framework to capture both per-request and per-connection information, and provide the first evaluation of classifiers for user action identification in HTTPS traffic. Such classifiers are important since proxies and middleboxes typically let HTTPS traffic pass through, and therefore do not have access to the individual HTTP requests carried over each TCP or TLS/SSL connection. Instead, we focus on classifiers that use the timing of new connection establishments or TCP/IP packet information to identify user actions. The accuracy of the classifiers is tested and evaluated against our ground-truth datasets. To understand the tradeoff between accuracy and the available information, we present a comparison of different (combinations of) classifiers as well as against a naive baseline approach (using the referer field).

Despite connections typically carrying many requests and our connection-based classifiers only associating a single request with each connection (i.e., the first over the connection), our analysis shows that our timing-based identification is still able to achieve surprisingly good accuracy. In fact, the per-connection based classifiers outperform our baseline classifier using the referer field. Even more encouraging are the results of the timing-based classifiers, which achieve most of the benefits of the advanced HTTP-aware classifiers.

II. Evaluation Framework

This paper presents results for a Firefox-based implementation and a simple browsing model. We have also implemented a framework in Chrome and tested more advanced browsing models. For a complete description of our methodology and results we refer to our full paper [16]. Code and scripts will be published with the full paper.

To allow automated ground-truth labeling and accurate evaluation of different user action identification classifiers, our testbed includes both client and proxy instrumentation. First, at the client side we collect ground-truth traces, which label each HTTP request based on whether it is due to a user action or not. We instrument both a Firefox and a Chrome client, to collect ground-truth traces of the user actions. At a high level, our trace collectors are implemented as browser extensions and record all HTTP requests made from the browser. Furthermore, event listeners are used to identify user actions and group requests.

Second, we use Squid proxy to collect information about all HTTP requests and their connection information (port numbers and IP addresses) as seen at the proxy. The Squid software[1] is installed on a proxy server between the client machine and the Internet. We disable caching, but set up the proxy to record information about all requests and responses, including timing information, server name, URL, content type, and other HTTP header fields. To allow connection-based analysis, we also capture port numbers and IP addresses.

Third, different request classifiers for user action identification (Section III) are applied on the proxy logs, so as to generate labeled traces based on each classifier's request classification rules, and the accuracy of each classifier is evaluated by comparing the labeled traces with the ground-truth trace. For this analysis, the primary evaluation measures are (i) precision, (ii) recall, and (iii) the $F1$-score. The precision is equal to the percentage of labeled user actions that in fact are user actions (in the ground truth). The recall is equal to the percentage of the total user actions (in the ground truth) that are correctly labeled (by the classifier) as user actions. Finally, the $F1$-score incorporates both precision p and recall r according to the equation $F1 = \frac{2pr}{p+r}$.

Finally, for further automation, we include an automatic web navigation tool that implements tunable user models that allow automatically generated ground-truth traces. Our automation script simulates a user's behavior while browsing the web according to a first-order continuous-time Markov process. After requesting an initial webpage, this model assumes exponentially distributed think times (i.e., the time until the next user action), after which the user probabilistically decides either to click on a link or request a new webpage. Controlling the think times and click probabilities (e.g., 0.85 as per the damping factor used in PageRank [3]) we can emulate different client behaviors: from impatient users with short average think times to users with longer think times.

III. User Action Identification

This section first describes three timing-based classifiers that use the timing of HTTP requests to identify user actions. We then show how these classifiers can be applied to HTTPS traffic. Finally, to better understand the potential limitations of user action identification in HTTPS logs, we also present HTTP-aware classifier extensions.

[1]Squid, www.squid-cache.org, May 2015.

A. Timing-based Classifiers

Previous (P): An HTTP request is considered a user action if the time since the most recent HTTP request is greater than a threshold T_{prev}. This classifier has been used by others (e.g., Liu et al. [13]) and is motivated by the observation that the time interval between two consecutive user actions typically is much greater than the time between consecutive HTTP requests for objects associated with a single page load.

Repetition (R): Many webpages periodically update their content, resulting in HTTP requests well after the page was originally loaded. To avoid misclassifying these requests, the *repetition* classifier filters out repeated requests if the same URL path has been requested more than K_{rep} times within time window T_{rep}.

Next (N): As the second HTTP request for a website does not occur until after the response of the first has been processed (and embedded objects have been identified within the data provided in the response), there is typically a time gap after the first (user action) request. The *next* classifier filters away HTTP requests that are followed by another HTTP request within a short time window Δ_{next}. This classifier attempts to aggregate user actions, while filtering bursts of requests associated with drag-along content.

Figure 1 shows a simple example scenario, illustrating the improvements of applying each of the three timing-based classifiers in successive combinations.

B. HTTPS Traffic and Connection-based Extensions

We consider two approaches to apply the above timing-based classifiers on HTTPS traffic. First, both Previous and Next can be applied directly on a per-connection basis. We use the time that a new connection is established as an estimate of the "lead" (object) request on that connection. Since this timing will be very similar to the timing of the (lead) HTTP requests themselves (tunneled within SSL/TLS, in the case of HTTPS) we can use similar thresholds as with the original Previous and Next classifiers. We do not consider any variation of the repetition classifier here. It should also be noted that user actions to a new webpage typically result in a new connection, while the majority of the (later) requests over an already open connection are due to non-user actions, and connections typically close before a website is re-visited again (e.g., the default keep alive timeout value of Apache servers is 5 seconds).

Second, more detailed packet inspection can be used to identify the timing of the later requests made over an already established (HTTPS) connection. In particular, we have observed that it is possible to extract the timing of most HTTP requests sent over the secure SSL/TLS connection by identifying the packets going from the client to the server that are larger than the packet size of a regular TCP acknowledgement (ACK) that does not include any payload. This allows us to distinguish packets that include HTTP requests from pure ACK packets. Again, without access to URLs, the best timing-based HTTPS classifier considered is Previous+Next.

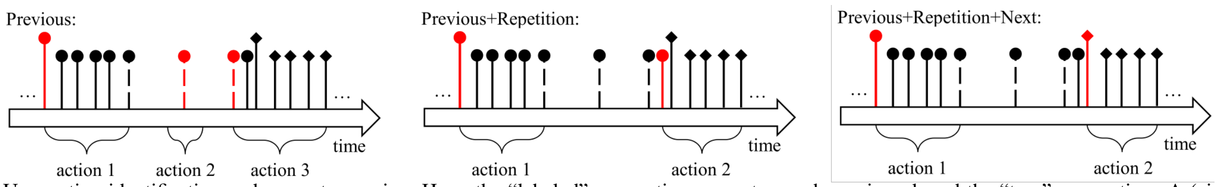


Fig. 1. User action identification and request grouping. Here, the "labeled" user-action requests are shown in red, and the "true" user actions A (circles) and B (diamonds) are shown using longer pin markers. Dotted markers represent repeated requests.

C. HTTP-aware Classifiers

To understand the value of HTTP-specific header information, we also consider additional filters.

URL (U): With our URL-based extension, requests are filtered out if the URL includes terms associated with frequently observed drag-along content [11].

Content (C): This classifier extension only keeps requests with content-type `text/html`. While the majority of user actions is of this type, the classifier works poorly on its own as much of the drag-along content also is of type `text/html`.

Referer: We have also considered the use of the referer field; both alone and in combination with other classifiers. However, we have found that referer-based classifiers in general perform poorly. For example, experiments with the *referer* classifier alone resulted in a precision of 0.27, a recall of 0.71, and an $F1$-score of 0.39. Some reasons for this are that the referer field often is left empty, is altered by the browsers, and the embedded elements themselves may generate requests.

IV. Classifier Evaluation

For the evaluation presented here, we collected five example traces. Each trace has 500 simulated user actions based on the simple browser model (Section II) and is generated with an average think time of 20s. Here, the pool of starting pages is randomly selected from the top-1,000 most popular webpages according to alexa.com, excluding HTTPS pages and Chinese websites (using non-Roman script). HTTPS pages were omitted to allow fair head-to-head comparison.

On average, each trace of 500 page requests resulted in 29,506 HTTP requests, distributed over 14,168 connections. Using our default thresholds ($T_{prev} = 2$s, $N_{rep} = 5$, and $\Delta_{next} = 20$ms), all timing-based classifiers (P, R, N) and their URL (U) and content (C) based extensions identify approximately 400 of the 500 user actions (matches), while only resulting in relatively few false positives (misses).

Figure 2 shows a breakdown of the matches and misses for all classifiers. Here, Previous (P) alone results in on average 404 matches and 276 misses, compared to the best classifier (P+R+N+C) that on average has 415 matches and only 35 misses. We can also see that the other two timing-based classifiers, Repetition (R) and Next (N), result in additional improvements when combined with Previous (P), and that the combined timing-based classifier (P+R+N) performs closer to the best classifier than Previous (P) alone.

Since the connection-based classifiers (P and P+N) are not able to identify HTTP requests over already open connections, it is not surprising that these classifiers identify fewer user actions (221 and 219, respectively). However, it is important to note that on average only 254 of the 14,168 connections were initiated due to user actions. The other connections were opened due to drag-along content. This shows that the connection-based classifiers do an excellent job identifying user actions, using only timing-based information, while keeping the false positives (misses) at a reasonably low level. In fact, all classifiers, including the connection-based classifiers, filter out (i.e., avoid to mislabel) 97% of the 29,506 HTTP requests (or 14,168 connections).

These results are very encouraging. First, the relatively simple connection-based classifiers achieve relatively good accuracy on their own. For example, the P+N classifier achieves an $F1$-score of 0.48 (calculated over requests), outperforming the referer classifier ($F1$=0.39), which has access to the HTTP header information of all requests. Second, the P+N classifier that identifies requests based on packet sizes (but does not try to identify repeat requests) leaves limited room for improvements, as it performs close to the classifiers that take into account content type and URLs, for example.

A. Classification Accuracy

The accuracy of the classifiers is relatively insensitive to the repetition (N_{rep}) and next window (Δ_{next}) threshold, but depends more on the previous window threshold (T_{prev}).

Figure 3 shows the relative tradeoff between precision and recall for the different classifiers. (Figure 4 shows Chrome results.) Here, T_{prev} is varied from 500ms to 10,000ms, $T_{prev} = 500$ms corresponds to the right-most point, and statistics are calculated over all requests, including requests invisible to the connection-based classifiers. Even though the threshold affects the classifiers differently, there is a clear ordering of the classifiers' identification abilities. For example, focusing on a fixed recall rate, each classifier improves the precision of the prior classifier.

Figure 5 shows the $F1$-score as a function of T_{prev}. For any threshold between 0.5-10s (the range used for Figure 5) the $F1$-scores for the timing-based classifiers are good (above 0.6 and 0.4, respectively) when using the timing of HTTP requests and the connection establishment. The non-overlapping 95% confidence intervals confirm that there are significant accuracy differences between the classifiers.

V. Related Work

Most research on user action identification relies on client-side or server-based techniques [5], [6], [9], [12], [18], [20]. Proxies and middlebox-based techniques, such as the classifiers discussed here provide a scalable alternative. However, because of their location, such techniques do not have explicit knowledge of user actions [1], [2], [14], [17]. Recent

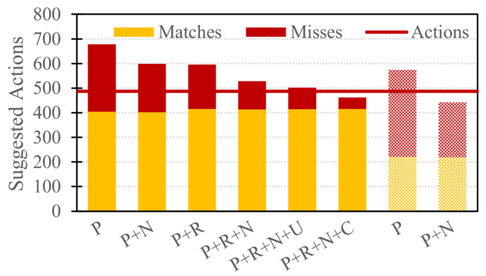
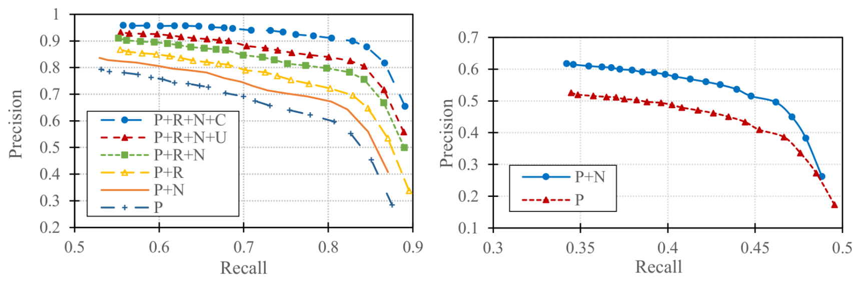


Fig. 2. Classification of requests labeled as "user actions", under default scenario. HTTP results (solid) on the left and per-connection results (shaded) on the right.



(a) Request based

(b) Connection based

Fig. 3. Precision-recall tradeoffs with previous window threshold as hidden variable.

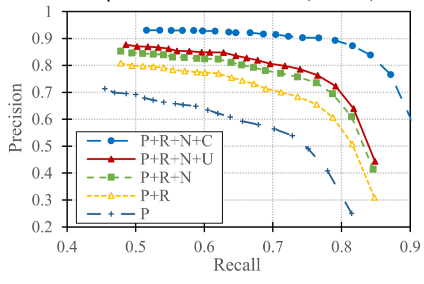
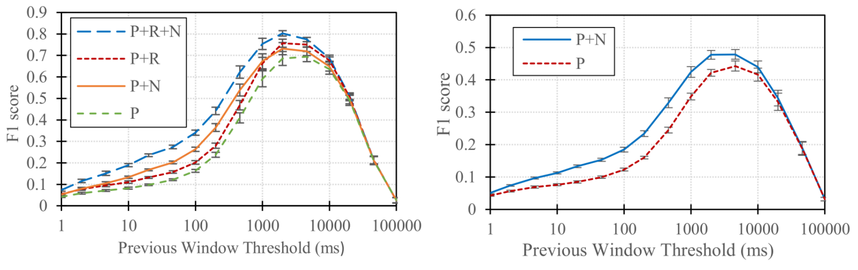


Fig. 4. Precision-recall tradeoffs with Chrome.



(a) Request based

(b) Connection based

Fig. 5. $F1$-score as a function of previous window threshold (T_{prev}) for the timing-based classifiers using the timing of HTTP requests and connection establishment, respectively.

works [8], [10], [19] utilize timing, content, and referer based heuristics. To evaluate these techniques, researchers have relied on manual classification for their ground truth.

Our work is orthogonal to these prior works. We present an automatic user action identification framework using browser extensions and automated browsing. We argue that manual labeling is time consuming, difficult, and extremely error-prone because of the large volumes of third-party drag-along content [4], [11], [15]. Prior work has also exclusively focused on user action identification of HTTP traffic. In contrast, we develop an automated evaluation framework, which includes automated ground-truth labeling, and show how the framework can be applied on both HTTP and HTTPS traffic. We are the first to evaluate user-action labeling of HTTPS data.

VI. Conclusions

This paper presents an automated evaluation framework for request classifiers for user action identification. The framework includes client ground-truth measurements through both Firefox and Chrome-based implementations, network-side proxy logs, and automatic tunable web navigation. The ground-truth traces can be used for head-to-head comparison of generic classifiers for user action identification, thus revealing the classifiers' accuracy. We also present the first evaluation of user-action labeling of HTTPS data. The results are encouraging, with connection-based classifiers identifying a significant portion of the user actions. Perhaps most encouraging is that the timing of HTTP requests can be approximated through careful packet size monitoring of HTTPS sessions and that our previous+next (P+N) classifier (that only use timing information) can achieve most of the benefits of the advanced HTTP-aware classifiers.

References

[1] A. Balachandran et al. Modeling web quality-of-experience on cellular networks. In *Proc. ACM MobiCom*, 2014.
[2] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proc. ACM SIGMETRICS*, 1998.
[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, Apr. 1998.
[4] M. Butkiewicz et al. Characterizing web page complexity and its impact. *IEEE/ACM Trans. on Networking*, Jun. 2014.
[5] F. Chierichetti et al. Stochastic models for tabbed browsing. In *Proc. WWW*, 2010.
[6] P. Dubroy and R. Balakrishnan. A study of tabbed browsing among Mozilla Firefox users. In *Proc. CHI*, 2010.
[7] P. Gill et al. Characterizing organizational use of web-based services: Methodology, challenges, observations and insights. *ACM Trans. on the Web*, Oct. 2011.
[8] Z. B. Houidi et al. Gold mining in a river of internet content traffic. In *Proc. TMA*, 2014.
[9] J. Huang and R. W. White. Parallel browsing behavior on the web. In *Proc. ACM HT*, 2010.
[10] S. Ihm and V. S. Pai. Towards understanding modern web traffic. In *Proc. ACM IMC*, 2011.
[11] B. Krishnamurthy and C. Wills. Privacy diffusion on the web: A longitudinal perspective. In *Proc. WWW*, 2009.
[12] C. Liu et al. Understanding web browsing behaviors through Weibull analysis of dwell time. In *Proc. ACM SIGIR*, 2010.
[13] J. Liu et al. Identifying user clicks based on dependency graph. In *Proc. WOCC*, 2014.
[14] B. A. Mah. An empirical model of HTTP network traffic. In *Proc. IEEE INFOCOM*, 1997.
[15] J. Purra and N. Carlsson. Third-party tracking on the web: A Swedish perspective. In *Proc. IEEE LCN*, 2016.
[16] G. Rizothanasis et al. Identifying user actions from HTTP(S) traffic. Technical report, 2016.
[17] F. D. Smith et al. What TCP/IP protocol headers can tell us about the web. In *Proc. ACM SIGMETRICS*, 2001.
[18] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *Proc. WWW*, 2007.
[19] G. Xie et al. Resurf: Reconstructing web-surfing activity from network traffic. In *Proc. IFIP Networking*, 2013.
[20] H. Zhang and S. Zhao. Measuring web page revisitation in tabbed browsing. In *Proc. CHI*, 2011.