Web Browsing Behavior Analysis and Interactive Hypervideo

LUIS A. LEIVA and ROBERTO VIVÓ, Universitat Politècnica de València

Processing user interaction data is well-known to be cumbersome and mostly time-consuming, specially when it comes to web browsing behavior analysis. Current tools usually display user interactions as mouse cursor tracks, a video-like visualization scheme that allows researchers to easily inspect what is going on behind the gathered data. However, to date, traditional online video inspection has not explored the full capabilities of hypermedia and interactive techniques. In response to this need, we have developed SMT2ϵ, a web-based tracking system to analyze browsing behavior using feature-rich hypervideo visualizations. We compare our system to related work in academia and industry, showing that ours features unprecedented visualization capabilities. We also show that SMT2ϵ efficiently captures browsing data, and is perceived to be helpful and easy to use. A series of prediction experiments illustrate that raw cursor data are both accessible and easily manipulable, providing evidence that the data can be used to construct and verify hypotheses. Considering its limitations, it is our hope that SMT2ϵ will assist researchers, usability practitioners, and other professionals interested in understanding how users browse the Web.

Categories and Subject Descriptors: H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Video, Evaluation/methodology*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Web-based interaction*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*Navigation*

General Terms: Experimentation, Human Factors, Design

Additional Key Words and Phrases: User interaction, information visualization, remote tracking, cursor behavior, hypervideo synthesis, interactive analysis, data mining, usability evaluation

ACM Reference Format:
Leiva, L. A. and Vivó, R. 0000. Web Browsing Behavior Analysis and Interactive Hypervideo. ACM Trans. Web 0, 0, Article 0 (0000), 27 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Investigating how User Interfaces (UIs) are operated has aroused historically a lot of interest in many research fields such as product design and software engineering; e.g., detecting areas of interest or misused layout spaces, time to complete a task, etc. To this end, video analysis has been considered a key component in multidisciplinary fields like Human-Computer Interaction (HCI) and Human-Centered Multimedia (HCM). It is important for practitioners to record what was observed, in addition to why such behavior occurred, and modify the application accordingly, if desired. Overall, observing the overt behavior of users has proved to be useful to investigate usability problems. For instance, based on live observations or analyses of video tapes, an evaluator can construct a problem list from the difficulties the users have accom-

This is a substantially extended version of a previously published conference paper [Leiva and Vivó 2012].
Work partially supported by the Spanish MEC/MICINN under the "MIPRCV Consolider Ingenio 2010" programme (CSD2007-00018) and the TIN2009-14103-C03-03 project.
Contact author's e-mail address: L. A. Leiva, llt@acm.org.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

plishing the tasks [Jacobsen et al. 1998]. Nonetheless, video data is time-consuming to process by human beings [Daniel and Chen 2003].

Analyzing video has traditionally involved a human-intensive procedure of recruiting users and observing their activity in a controlled lab environment. Therefore, several factors have led to the development of remote activity tracking for such a UI evaluation; for instance: *a)* relevant data of real world usage is required; *b)* it is difficult to gather a representative user sample; *c)* rapid prototyping requires sometimes preliminary studies and time is a restriction; *d)* recruiting users to a lab can be very costly, including equipment and resources; *e)* user's work context is difficult or impossible to reproduce in a laboratory setting; *f)* software applications usually have a life cycle extending well beyond the first release; and *g)* analytical evaluation of systems should complement and, sometimes, extend empirical methods.

Processing user interaction data is therefore, at a minimum, cumbersome. Moreover, if we move to the Web domain, analysis is, at best, fairly limited and, at worst, virtually impossible. At present, the Web has grown massively in size, popularity, applications, devices, and number of users. The Web is so popular that usability on this domain has more impact on the society than some others have [Chen et al. 2001]. Unfortunately, to date most theories on browsing behavior are based on the study of patterns from server access logs. Such patterns consist of click-paths or query logs; i.e., sequences of browsed URLs. This may be enough for quantifying some observations, such as analyzing the so-called conversion rates and detecting whether the website has succeeded or failed to reach their goals. However, the context of actions is of utmost importance to understand browsing usage. Thus, when facing a finer-grained exploration of human interaction on the Web, analytics based on server access logs alone are anything but accurate, being necessary to move towards the client side.

State-of-the-art user tracking systems employ JavaScript logging tools, which fundamentally include mouse and (sometimes) keyboard tracking, since these input devices are ubiquitous. This way, it is not required specific hardware or special settings to collect interaction data remotely on the client side. Furthermore, researchers have shown that the mouse cursor can inform about user intent and search interests, finding a strong correlation to how likely a user will look at web pages [Chen et al. 2001; Mueller and Lockerd 2001; Huang et al. 2011]. Overall, remote activity tracking provides a series of interesting advantages when compared to classical usability tools. According to Arroyo et al. [2006]: *1)* it can be mass deployed, allowing for large datasets; *2)* it is able to reach typical users and first time visitors in their natural environment; *3)* it can continuously test live sites, offering new insights as new website sections are deployed; and most importantly, *4)* it is transparent to the users, so no experimenter bias or novelty effects are introduced, allowing users to browse websites as they would normally do.

Current tracking systems usually support replaying user interactions as mouse cursor tracks, a video-like visualization scheme that allow researchers to easily inspect what is going on behind such interactions; e.g., *How many of the users considered clicking on the "Buy" button, and how many of them did actually click on it? In which order did the user fill in the form fields? Do users ever scroll the web page? If so, how far exactly?* However, traditional online video inspection has not benefited from the full capabilities of hypermedia and interactive techniques. We believe these capabilities can better assist both novel and expert usability practitioners. On the one hand, for a long time hypermedia systems have been demonstrated to be engaging tools that make complex information understandable, handling images and motion as representational aids [Shiffer 1995]. On the other hand, interactive visualizations do support the comprehension of human behavior in both qualitative and quantitative ways (e.g., characterizing changes in activity over time). Yet efficiently implementing this technology is a challenging task, as making video interactive and linking video and text has traditionally posed a series of general problems [Liestøl 1994].

It is our belief that hypervideo will gain notable importance in the web tracking domain. Concretely, we believe practitioners and researchers—from here onwards we will refer to these figures as "the viewer"—can be presented with cursor data in a more engaging way. We also believe that, in addition, new ways of dealing with video-based cursor data demand new ways of interaction. With these ideas in mind, we developed SMT2ϵ, a general purpose tracking tool for understanding web browsing behavior. SMT2ϵ is a significant continuation of our previous work, SMT2 [Leiva and Vivó 2012], with a solid focus on better recording and hypervideo-based visualization capabilities.

1.1. Revisiting Hypervideo

Hypervideo, or hyperlinked video, is a video stream that contains user-clickable anchors, allowing navigation between video and other hypermedia elements [Smith and Stotts 2002]. As such, hypervideo offers the possibility to change from the linear order found in current cursor replay systems to a non-linear and feature-rich visualization.

We redefine hypervideo as *a composite video stream that contains embedded interactive elements*, which is a more general definition yet compatible with the literature [Hirata et al. 1993; Francisco-Revilla 1998; Smith and Stotts 2002]. What differentiates SMT2ϵ from other cursor replay systems, though, and from classical hypervideo systems, are the following manipulation capabilities:

— Different visualization choices can be made based on a series of informative layers that are rendered at runtime. This way, the content of hypervideos is modified according to the viewer's interests in real time.
— Multiple browsing sessions can be combined in a non linear structure, thus allowing the viewer to save time.
— Hypervideos can be linked to specific frames using timecode format (hh:mm:ss), enabling, e.g., sharing a specific time segment with friends or co-workers.
— Hypervideos do support HTML-based annotations, which allows the viewer to add content with pointers to time segments; e.g., to mark interesting parts for later review or even link to other hypervideos.
— Interaction profiles are generated as soon as new users access the website. Therefore, new data are available to analyze immediately upon installing the system.

In addition, SMT2ϵ can be further augmented with JavaScript plugins and PHP extensions, so, being web-based, additional functionality can be easily implemented. Finally, it is worth pointing out that this tool is released as open source software, which encourages collaboration, code fixes, bug reports, and other facilities that are not available in proprietary systems.

1.2. Organization

This paper is organized as follows. First, in Section 2 we review previous work that relates to our system, and provide an overview of SMT2 and how SMT2ϵ differs from it. Next, in Section 3 we examine a series of questions about usage that can be answered by visualizing mouse cursor replays. The relevant parts of our system are thoroughly described in Section 4. Further on, empirical evaluations are performed in Section 5, which mainly comprise *(1)* a comparison of our tool against a commercial system under two studies (logging performance and usability evaluation); and *(2)* a series of experiments in which we predict user interaction, such as browsing time or scroll reach, using mouse cursor features alone. We provide a general discussion in Section 6, and acknowledge the limitations of our system in Section 7. Finally, we close the paper with a series of concluding remarks and opportunities for future work.

2. RELATED WORK

The utility of user tracking systems is evident through the research literature. For instance, mouse cursor data has been recently used to conduct studies on eye-mouse coordination [Rodden et al. 2008; Huang et al. 2011; Huang et al. 2012], web search [Agichtein et al. 2006; Guo and Agichtein 2010a; Guo and Agichtein 2012], reading behavior [Guo and Agichtein 2010b; Hauger and Van Velsen 2009; Hauger et al. 2011], or user modeling [Leiva and Vidal 2010; Leiva 2011; Buscher et al. 2012]. Yet to date the vast majority of research studies are conducted using *ad-hoc* scripts. Therefore, a unified system to capture and manipulate such an interaction data seems appealing.

In this section, given the focus of the paper, we relate to remote user tracking systems from industry and academia. Note that earlier approaches involved installing software on the client-side [Reeder et al. 2001; Tarasewich et al. 2005]. Instead, here we comment on approaches that do not need to do so, since it is the way most systems operate at present.

2.1. Web Tracking Systems in Research

We found [Mueller and Lockerd 2001; Arroyo et al. 2006; Atterer et al. 2006] to be the contributions among the research community that relate most to our work. These systems, though currently seem to be no longer maintained, have set the precedent in web-based user tracking applications, both in academia and industry. They were released as open source software, so they can be easily inspected. Although not stated, Mueller and Lockerd [2001] presumably collected cursor movements in the same way as Arroyo et al. [2006], i.e., storing in a database $\{t, x, y\}$ tuples (timestamp, coordinates) whenever the mouse moves out of a R px circle radius. Arroyo et al. [2006] introduced the concept of collaborative filtering (that is, working with aggregated users' data), and also the idea of using a web-based proxy to track external websites. Finally, Atterer et al. [2006] developed an advanced HTTP proxy that tracked the user's every move, including keystroke data, when a periodical scroll or a cursor event was captured. Interaction data were saved in a textual format akin the Apache access logs[1], and the system could map cursor coordinates to Document Object Model (DOM) elements. While [Mueller and Lockerd 2001; Arroyo et al. 2006] overlaid an image on top of the web pages to represent the tracked interaction data, in [Atterer et al. 2006] it also appeared feasible to replay the actions of a particular user, though visualization was not the primary focus of the system. We argue that incorporating the temporal information may enhance human interaction understanding, to replay exactly how users interact on a website. To this end, this is where video capabilities come into play, which, to some extent, have been lately implemented in industry systems.

2.2. Web Tracking Systems in Industry

We inspected the code of the most popular commercial web tracking systems via reverse engineering, which was possible by using their freemium versions—luckily, most of them offer a limited but functional service free of charge via email subscription. Among the available candidates, we found different approaches to register and visualize user interaction. Therefore, we begin highlighting the main differences while recording. Then, we shall compare each candidate with ours in terms of visualization.

Basically, commercial systems work as "hosted solutions", i.e., a software-as-a-service delivery model. These systems require the webmaster to insert a tracking script in the pages to be targeted. Then such a tracking script transmits the data back to the commercial server(s). Eventually, registered users can review the tracking logs at an administration area or "admin site" provided by the commercial system.

On the one hand, CLICKTALE[2] has a very active development at present, and is deeply oriented to web analytics. Among other setup instructions, it requires the user to establish a recording ratio, to determine the percentage of visitors that will be selected. We use random sampling instead, that is, assigning equal probability of selection for every user. Nevertheless, we let this feature to the webmaster's discretion, allowing him to establish any desired sampling strategy (see Figure 5). To enable cross-domain communication, CLICKTALE uses the strategy depicted in [Atterer et al. 2006], i.e., requesting an image having a query string with the tracked data in the `src` attribute via AJAX. USERFLY[3] and MOUSEFLOW[4] are also popular cursor tracking tools, both built on top of jQuery (a general-purpose framework for JavaScript development). USERFLY encodes tracking data in a JSON string, while MOUSEFLOW uses base64url format. Then data are sent in the same way as in [Atterer et al. 2006]. We set out later our approach for transmitting data, and how does it differ from these systems.

On the other hand, MPATHY[5] uses Flash socket connections to send the data whenever a browser event is registered. CLIXPY[6] creates a log file in a very similar way as Atterer et al. [2006] did, i.e., sequences of timestamped events with a corresponding identifier string, with the difference that logs do not explicitly include the user IP. This system sends tracking data to a Flash object each 3 seconds to allow cross-domain requests. In both cases, relying on the Flash plugin on the client side poses a fundamental problem, since some users do not have such plugin installed, either because it is not supported on their operating systems (e.g., iOS) or because simply they do not want it, so it would not be possible to track those users at all. Therefore, depending on the target audience of the website, it could lead to missing a huge fraction of the visitors that otherwise would have provided valuable insights about their browsing experience.

Regarding visualization, the conceptual idea used in our tool is similar to the ones used in the majority of the above-mentioned commercial systems: rendering logged data events on top of an HTML page. However, current systems, besides of providing limited support in terms of visualization capabilities, only allow to reproduce a single user session at a time. This can be extremely time-consuming, depending on the number and duration of the visits. For that reason, we let the viewer to merge simultaneously as many interaction logs as she would like. Additionally, SMT2ϵ allows to modify the information to be displayed at runtime, so that it can ease the process of testing different visualization strategies (Figure 6). Furthemore, SMT2ϵ provides the viewer with actual hypervideos, enabling annotations and links to specific parts of the visualization.

Finally, there exist other approaches for visualizing user's activity, such as DOM-only based (see, e.g., the <TAG> tracker[7] system), or heatmap-based. The former is lately gaining support in general-purpose web analytics software. The latter, on the other hand, is implemented to some extent by most of the above-mentioned commercial tracking systems. Our tool supports both types of visualization, but the way we process the data is, again, more focused on infographics instead of on web analytics. For instance, our heatmap visualization strategy is *dynamic*, allowing to generate the maps at runtime, i.e., at the same time the viewer is watching the hypervideos. Specifically, the resulting heatmap follows the 'shadowmaps' implementation detailed in [Špakov and Miniotas 2007], and can be seen in Figure 6d.

Besides of the technical comparisons described below, analyzing interaction data like current tracking systems do can be rather limited; as the viewer is able to use at best play/pause as playback controls while replaying a user session. Now we shall describe

SMT2, our previous work, and show how it differs from current systems. Then, we introduce SMT2ϵ, and describe how it differs specifically from SMT2.

2.3. SMT2 Overview

Firstly, an important feature of our previous work regarding to state-of-the-art web tracking systems is the ability of compositing multiple interaction logs into a single hypervideo. This feature is useful to assess qualitatively the usability of websites, and also to discover common usage patterns by simply inspecting the visualizations (see Section 3).

Secondly, another important feature of SMT2 is the generation of user and page models based on automatic analysis of collected logs. In this regard, we did not find any related tracking system that would perform implicit feature extraction from users' interaction data; i.e., interaction features inherently encoded in cursor trajectories. We believe this is a promising line of research, and currently is gaining attention from other authors; e.g., [Guo and Agichtein 2010a; Huang et al. 2011].

Thirdly, the recording approach used in SMT2 is different regarding the ones described in Section 2.2. Concretely, it performs a discretization in time of user interactions, following a simple event logging strategy and the *polling* technique; i.e. taking a snapshot of the cursor status (mainly coordinates, clicks, and interacted elements) at a regular interval rate. This way, SMT2 tracks user actions as they were exactly performed. This also allows to modify the speed at which movies can be replayed, which is helpful to normalize trajectories that were acquired at different sampling rates when compositing a multi-track hypervideo.

2.4. Introducing SMT2ϵ

Regarding tracking capabilities, SMT2ϵ behaves almost identically as its predecessor, with the notable exception that SMT2ϵ features LZW compression to transmit logged data, saving thus bandwidth. This capability is evaluated in Section 5.1.1. The actual improvements made to SMT2 that eventually derived in SMT2ϵ are focused on the server side.

First of all, our current effort goes towards interactive hypervideo synthesis from user browsing behavior. However, unlike conventional hypervideo, SMT2ϵ is aimed to build full interactive movies from remotely logged data. Furthermore, current hypervideo technology itself is limited to clickable anchors [Smith and Stotts 2002]. SMT2 augmented this technology with *interactive* **infographics**, i.e., a series of information layers that are rendered at runtime and provide the viewer with additional information. For instance, hovering over a click mark displays a tooltip indicating the cursor coordinates, or hovering over a hesitation mark displays the amount of time the cursor was motionless. SMT2ϵ extends this hypervideo-based technology with: (1) **hyperfragments**: visualizations can be linked to specific start/end video parts, and (2) **hypernotes**: HTML-based annotations that point to specific video parts.

We believe these novel improvements are convenient in a cursor tracking visualization system for a series of reasons. First, hyperfragments allow the viewer to select a "slice" of the video that may be of particular interest. Hyperfragments can be specified either with a starting or an ending timecode. When doing so, the video length is trimmed to such specified duration. This lets viewers quickly access desired information without having to review the entire replay. Second, hypernotes allow the viewer to comment on the visualizations at a specific point in time; e.g., to annotate some details about the video or let co-workers know that such video has been reviewed. When a hypernote is created, the viewer can click later on a note icon on the timeline that will seek the replay to the creation time of the hypernote (Figure 6a). This provides the viewer with indexing capabilities that can be extended to content searching.

(a) (b) (c)

Fig. 1: Combining visualization possibilities. [1a] Displaying hesitations (circles) and clicks (small crosses). [1b] Displaying entry/exit coordinates (cursor bitmaps), motion centroids (crosses), drag&drop activity (shaded fog), and interacted DOM elements (numbered rectangles). [1c] Analyzing a decision process; the user rearranged items in a list. Small circles represent cursor dwell times. Hovered DOM elements are labeled based on frequency (percentage of browsing time), including a blue color gradient (100% blue: most hovered items). The same scheme is used to analyze clicked items, but using red color.

Fourth, hypernotes are HTML-based, which enables rich-formatted text and insertion of images and links (for instance, to point to other hypervideos). This capability opens a new door to how visualizations can be later processed; e.g., it would be feasible to summarize a user session in a textual format.

In addition, SMT2ϵ features two installation modes: as an all-in-one solution (when website and admin site are both placed in the same server) and as a hosted service (website and admin site are placed in different servers). SMT2 was limited in this regard, since to allow cross-domain communication, every website would require at least PHP support to forward the requests to the storage server (i.e., the admin site). With SMT2ϵ, however, the only requirement for a website to be tracked is inserting a single line of JavaScript code, as other commercial systems do, so potentially any website can use it.

Finally, SMT2ϵ allows the viewer to classify pages according to user behavior (which is known as *behavioral clustering*) in real time, by automatically mining cursor features from the database logs. The inclusion of this functionality was motivated by the fact that users may find it useful to discover common interaction profiles as well as to easily identify outliers as new users access the website.

3. APPLICATIONS

To begin with, we provide a systematic examination of some questions about usage that can be answered by visualizing mouse cursor replays. As shall be hinted, currently most of these questions can only be answered by using our tool. Yet in Section 3.2 we enumerate a series of scenarios where any user tracking system could suitable fit.

3.1. Decoding Browsing Behaviors

— ***Where do users hesitate? How much?*** We followed the notion of *dwell time* introduced in [Müller-Tomfelde 2007], also named *iddle time* [Buscher et al. 2012], which is essentially the time span that people remain nearly motionless during pointing at objects. Cursor dwell times are usually associated with ambiguous states of mind [Arroyo et al. 2006], possibly due to a thinking or cognitive learning process. In SMT2ϵ, dwell times are displayed as circles with a radius proportional to the time in which the cursor does not move (Figure 1a). The system takes care of not displaying extremely large values of dwell times, by limiting the circle radii to a quarter of the viewport size. Currently, dwell times are handled similarly in other cursor tracking tools such as CLICKTALE.

— ***Do users perform drag&drop operations? How?*** Users perform drag and drop to select HTML content, or also to download an image to their desktop or to a file manager window. At a higher level, a web application can support rearranging widgets to customize their layout, or also by adding objects to a list to be processed. SMT2ϵ uses a specific format to encode cursor coordinates, borrowed from the handwriting community, where the status of the click button is attached to each cursor coordinate. Therefore, SMT2ϵ provides a specific visualization type for these cases (e.g., Figure 1b). To our knowledge, this capability is only provided by our tool.

— ***What element is the user actually interacting with?*** Thanks to the bubbling phase of JavaScript events, whenever a cursor event is dispatched (e.g., `mouseover`, `touchmove`) the tracking script traverses the DOM hierarchy to find if there is an element that relates to the event. Each tracking log stores a list of interacted DOM elements, sorted by time frequency (Figure 1c), including hovered and clicked/tapped elements. Thus, such list can be inspected either quantitatively (by looking at the numbers) or qualitatively (by looking at the colors). It is worth mentioning that most web analytics applications only provide click information, since this information can be derived from server logs alone. Therefore, this visualization can be helpful to answer in-page browsing questions, such as if the users go straight to the content or whether the cursor hovered over a link without clicking.

— ***Which areas of the page do concentrate most of the interaction?*** To answer this question, a K-means clustering of the coordinates is performed each time a mouse track ends replaying. So, focusing on the clustered areas allows to visually notice where users are performing most of their actions. Each cluster is represented by a circle with a radius proportional to the cluster population (Figure 6c). This visualization layer is notably appropriate when tracking data are rendered as a static image. To date, SMT2ϵ is the only system that features this capability.

— ***Do different mouse tracks correlate?*** The viewer can select the 'time charts' option from the control panel (Figure 7) and visually compare multiple tracks (see Figure 2a). For instance, dwell times will be plotted as horizontal lines, and scrolling will be plotted as near-vertical lines. Each tick in the x-axis corresponds to the registration frequency used while tracking (e.g., for 24 Hz, each tick would be 1/24 seconds). Optionally, SMT2ϵ can display the average cursor trail for a given group of visitors. This feature may be of special interest when people behave similarly; all in all it provides a general gist of how users browse a page. These visualizations are also exclusively offered by SMT2ϵ.

— ***What is the persistence of the page?*** It is commonly agreed that, to some extent, page relevance is correlated to the time spent on browsing it. In this case, a 3D visualization (Figure 2b) might be useful to unveil these observations. The 3D chart plots the temporal evolution of cursor coordinates along the z axis, and provides simple interactive controls to ease further inspection. This way, it may be interesting to aggregate all visits for a given page and observe at a glance how long are cursor trails or if there exists some correlation between them. To date, only SMT2ϵ provides this type of visualization.

3.2. Envisioned Usage Scenarios

We believe that, beyond interactive visualizations, the scope of cursor replay systems in general, and SMT2ϵ in particular, can go further. In short, some real-world usage scenarios where these systems could suitably fit include the following:

(a) Normalized Y coordinates against time

(b) Interactive 3D visualization

Fig. 2: Time charts visualization. Bold line is the averaged cursor trail, which is an optional visualization feature that takes into account aggregated logs. In this example, there are three visits, and browsing times can be visually compared at a glance. The 3D view allows rotating the axes with 3 sliders (one per axis), zooming, and projecting the lines in the YZ, XZ, and XY planes.

Interaction Research. In general terms, understanding human movement is fundamental for improving input devices and interaction techniques.
Data Mining. Data samples can be obtained at a large scale, so one can perform readily prospective studies.
User Modeling. Cursor data can be used to build behavioral models with meaningful conceptual interpretation, according to user interactions.
Usability Testing. For a long time, activity tracking has been a reliable source of information about user interaction behavior.
Gesture Recognition. Gathered cursor positions can be mapped to specific gestures, allowing thus to trigger specific browsing commands on a given page.
Performance Evaluation. Compare motor skills or pointing abilities of different users (or groups thereof) within a particular task on a web UI.
Usage Elicitation. If we want to avoid possible biases while analyzing human usage patterns, then we must deal only with interaction data of real users.
Collaborative Filtering. Unveil usage profiles and statistics among multiple viewpoints, data sources, and so on.
Self-Adaptating UIs. Employ interaction data to modify the appearance of the page elements based on each user needs.

One may note that the applications depicted in the list above are generic enough to ensure a broad generalization scope.

4. SYSTEM DESCRIPTION

SMT2ϵ has been built using web technologies and hence does not need to install additional software on the client side. The only requirement is a web browser with JavaScript support, so any modern device capable of accessing the Internet (e.g., laptops, smartphones, tablets, etc.) could be a fair candidate to take part in a tracking campaign. As described below, SMT2ϵ is composed of three fundamental parts: recording, management, and visualization. On the server side, any web server supporting PHP and MySQL can run both the admin site and the visualization application.

4.1. Architecture

This system uses the WWW infrastructure to log the user activity in a MySQL database (Figure 3). The technology used to create such an interactive movies is a mixture of PHP (to query the database), HTML (to overlay the tracking data on top of it), JavaScript (to process the aforementioned tracking data), and ActionScript (to build the hypervideos).

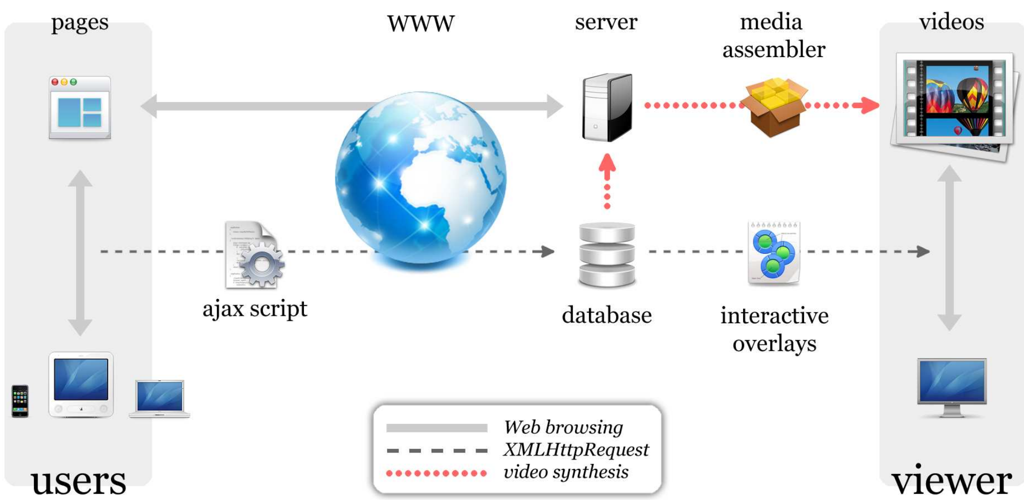


Fig. 3: System architecture for acquiring users' activity and synthesizing interactive hypervideos.

4.2. Logging Users' Interactions

While users are browsing as they would normally do, an Ajax script logs their interactions in the background and send the data back to the server at fixed-time intervals (Figure 4). Tracking is performed in a transparent way for the users, either silently or by asking their consent. SMT2ϵ uses the UNIPEN format [Guyon et al. 1994]—a popular scheme for handwriting data exchange and recognizer benchmarks—to store the mouse coordinates. This way, it is possible to re-compose the user activity in a fine-grained detail.

In order to reduce the data size to be transmitted, recording can be *continuous* (default behavior) or *intermittent* (i.e., tracking stops/resumes on blur/focus events), let-

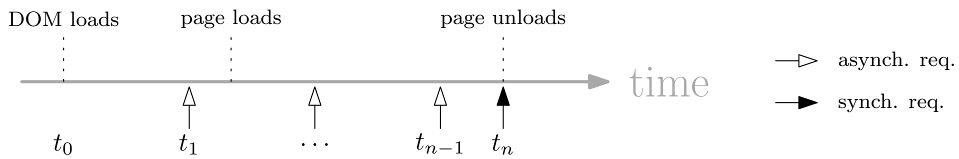


Fig. 4: Logging timeline example. Tracking data are asynchronously submitted at regular time intervals, until the page is unloaded (in which case the submission is synchronous, for cross-browser support).

ting the webmaster decide which operation mode is best suited to their needs. For instance, if an eye tracker is going to be used together with our system, then it is preferable to set continuous recording, in order to keep mouse and eye coordinate streams synchronized. On the contrary, if SMT2ϵ is going to be used alone, then the webmaster may want to save storage space in the database by enabling intermittent recording. This also results in less data transmitted to the web server. Also, as pointed out in Section 2.4, SMT2ϵ features LZW compression on the logged data, which contributes even more to save bandwidth.

Additionally, it is possible to store interaction data from different domains into a single database, provided that each domain is under the webmaster control. Otherwise, SMT2ϵ can fetch external websites by using a PHP proxy; i.e., users must start browsing from a dedicated proxy page and the system automatically inserts the required tracking code.

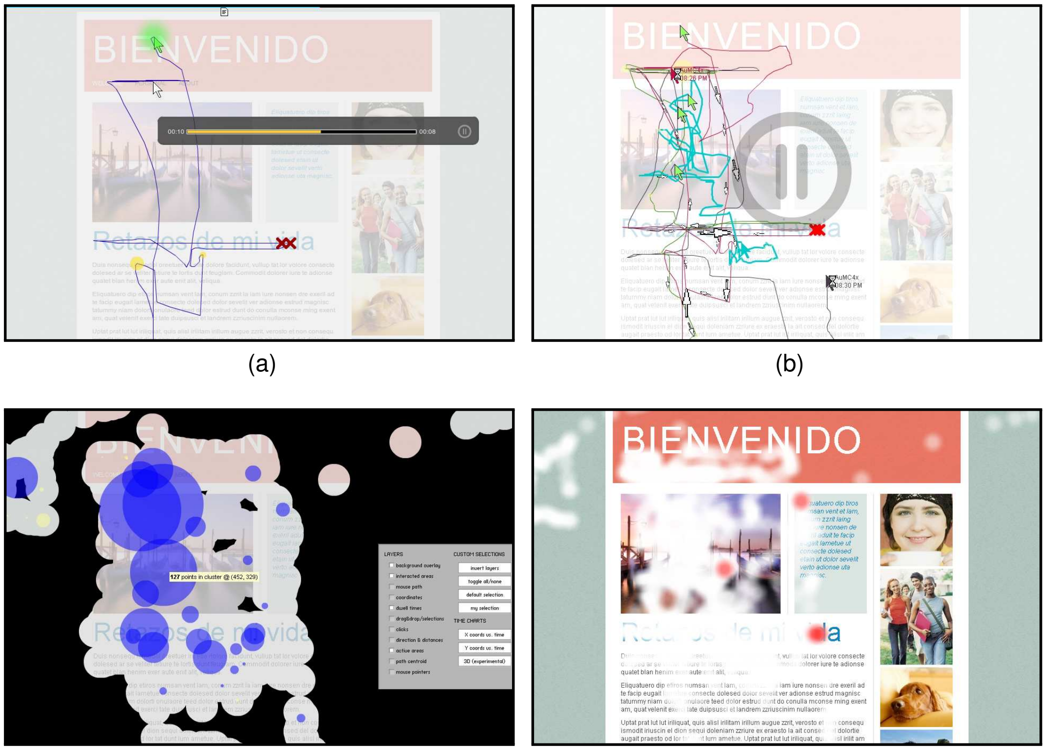
```
<script type="text/javascript" src="smt2e.js"></script>
<script type="text/javascript">
smt2.record({
  fps:          24,
  postInterval: 30,
  disabled:     Math.round(Math.random()),
  warn:         true,
  recTime:      180
});
</script>
```

Fig. 5: A working example of inserted tracking code. Registration frequency is set to 24 Hz (default value). Data will be sent at intervals of 30 seconds (default value). We also set random sampling for user selection, and ask consent to the chosen users for monitoring their browsing activity (they must agree to start recording), who will be tracked at most for 3 minutes.

4.3. Video Synthesis

The viewer indicates the information that will be pulled out from the database. For example, she might request to visualize a single browsing session (Figure 6a), in which case she just would click on a play icon from the 'user logs' list (Figure 14). Then, the system would retrieve the subsequent logs to replay all tracks sequentially. On the contrary, the viewer might want to filter out logs by operating system and browsed page, in which case she would use a 'mine results' form (Figure 14) that would merge the data into a single hypervideo (Figure 6b).

Retrieved data are encoded in JavaScript Object Notation (JSON), and fed into a precompiled video template. Cursor trajectories are normalized according to the original viewport of the user's browser and the current viewport of the viewer's browser,



(a) (b)

(c) (d)

Fig. 6: Some examples of our hypervideo visualization tool. [6a] single session with embedded media player and a couple of hypernotes attached to the timeline at the top; [6b] Replaying users' trails simultaneously, highlighting the average cursor trail and overlaying direction arrows. [6c] clusters of mouse movements, displaying also masked areas of activity; [6d] Dynamic heatmaps of mouse coordinates and clicks.

following a non-uniform affine mapping, either by scaling or translating the coordinates, depending on the type of layout: namely *left*, *center*, *right* or *liquid*. A cached copy of the browsed page and the above-mentioned interaction data are both bundled in the hypermedia player. This way, movies can be replayed within any web browser.

4.4. Accessing, Analyzing, and Interacting with the Data

The viewer can toggle different information layers interactively while she visualizes the videos by means of an embedded control panel (Figure 7).

Furthermore, movies can be generated for individual users or by taking into account different kinds of segmentations; e.g., time or date intervals, city locations, first-time users, and so on. For instance, the viewer can segment the tracking logs by user ID, and determine which elements were interacted most, or examine the percentage of scroll to infer interest; e.g., if all browsed pages of some user have minimum scroll reach, it may indicate that the user is searching for a specific page with no success. Figure 8 displays a screenshot of the analysis module.

Automatic analysis of interaction features is also feasible for mining patterns within the admin site, since collected data are readily available in the database. This way, besides explicit metadata that is assigned to content, implicit knowledge can help to

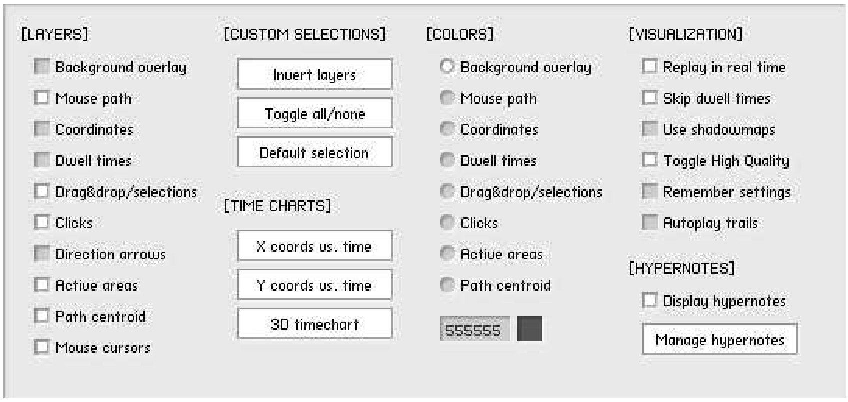
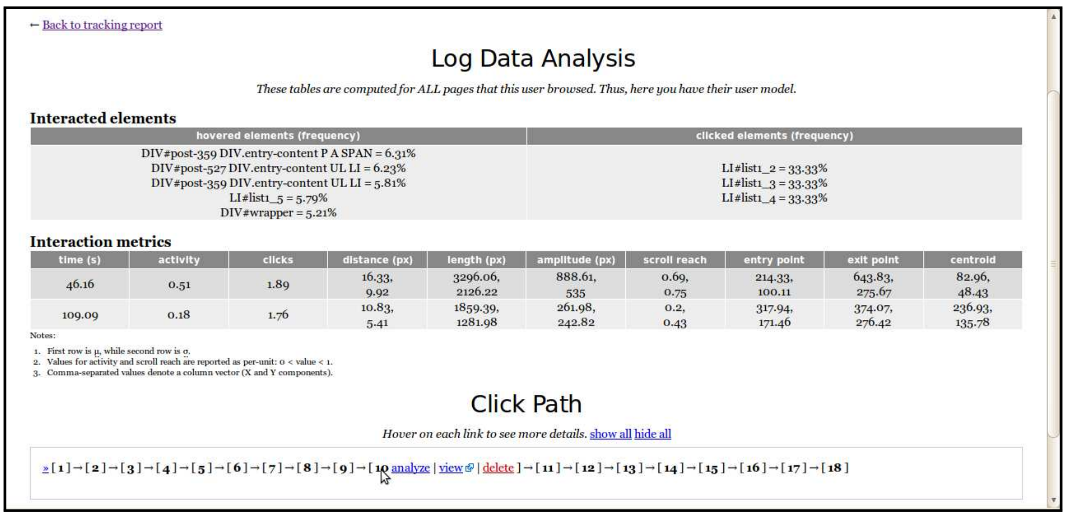


Fig. 7: A draggable control panel is the main link between the viewer and the synthesized hypervideos. One can manipulate different visualization possibilities, which will be applied at runtime.



← Back to tracking report

Log Data Analysis

These tables are computed for ALL pages that this user browsed. Thus, here you have their user model.

Interacted elements

hovered elements (frequency)	clicked elements (frequency)
DIV#post-359 DIV.entry-content P A SPAN = 6.31% DIV#post-527 DIV.entry-content UL LI = 6.23% DIV#post-359 DIV.entry-content UL LI = 5.81% LI#list1_5 = 5.79% DIV#wrapper = 5.21%	LI#list1_2 = 33.33% LI#list1_3 = 33.33% LI#list1_4 = 33.33%

Interaction metrics

time (s)	activity	clicks	distance (px)	length (px)	amplitude (px)	scroll reach	entry point	exit point	centroid
46.16	0.51	1.89	16.33, 9.92	3296.06, 2126.22	888.61, 535	0.69, 0.75	214.33, 100.11	643.83, 275.67	82.96, 48.43
109.09	0.18	1.76	10.83, 5.41	1859.39, 1281.98	261.98, 242.82	0.2, 0.43	317.94, 171.46	374.07, 276.42	236.93, 135.78

Notes:

1. First row is μ, while second row is σ.
2. Values for activity and scroll reach are reported as per-unit: $0 < \text{value} < 1$.
3. Comma-separated values denote a column vector (X and Y components).

Click Path

Hover on each link to see more details. show all hide all

» [1] → [2] → [3] → [4] → [5] → [6] → [7] → [8] → [9] → [10 analyze | view | delete] → [11] → [12] → [13] → [14] → [15] → [16] → [17] → [18]

Fig. 8: A log analysis example, reporting all visited pages for a particular user. The first table summarizes the most interacted DOM elements (hovered and clicked, respectively). The second table computes interaction features based on cursor activity (e.g., distances, entry/exit coordinates, etc.) In this table, the first row are mean values, and the second are standard deviations. Whether appropriate, values are decomposed in a 2-d vector, where dimensions are x and y coordinates. The click path is the sequence of pages that the user has browsed. Each item in the click path allows performing three actions over its corresponding log file: visualize, analyze, or delete.

get a better picture on the nature of such content (see Section 5.2.1). Concretely, the features that SMT2ϵ computes for a given tracking logs are described as follows.

Time. Browsing time (in seconds) spent on the page.
Activity. Fraction of browsing time in which the cursor was moving, defined in $[0, 1]$. (0: no movements at all, 1: otherwise).
Clicks. Number of issued mouse clicks.
Distance. Averaged L2 distance (in px) between coordinates.

Length. Cumulated sum (in px) of cursor distances.
Range. Difference (in px) between maximum and minimum coordinates.
Scroll reach. Percentage that informs how far did the cursor scrolled the page, defined in $[0, 1]$. (0: no scroll at all, 1: scroll reached the bottom of the page).
Entry/exit points. The first and last mouse coordinates, respectively.
Centroid. Geometric center of all coordinates.

4.5. Comments on an Early Prototype

We released a preliminary, beta version of SMT2 two years ago at the GoogleCode repository, and, overall, reaction to the system has been overwhelmingly positive. Since then, it has received successful feedback by researchers, developers, and practitioners worldwide.

Some users have reported useful features, either related to the visualization module or to general usability guidelines. Other users have suggested enhancements to the admin site. Developers have confirmed that SMT2ϵ could work with other programming environments, or have reported modifications that fulfilled their needs. Finally, other users have asked for further applications of the tool (e.g. in marketing research or web search tasks).

All in all, received feedback did help us to fix existing bugs and refine that preliminary prototype. We have incorporated as many suggested enhancements as we could, and currently SMT2ϵ is in a stable iteration. Some of the improvements included new visualization layers (e.g. dynamic heatmaps or the 3D visualization), the multi-user admin site with support for different roles, the PHP proxy integration to track external websites, and the automatic analysis of single/aggregated user and page models, among other revisited features. A detailed list of source code changes is available at http://code.google.com/p/smt2/source/list.

5. EVALUATION

The following is an empirical assessment of which we believe are the key properties of SMT2ϵ. In particular, we address *1)* logging performance (What is the network overhead incurred by tracking? How does frame rate influence in this regard?) *2)* system usability (Is the system easy to use? What is the user workload?), and *3)* utility of user modeling features (Can we predict interaction by using features derived from cursor data? Are these predictions reliable?). All experiments were performed on a Linux laptop with an i386 dual core processor @ 1GHz and 2GB RAM.

Regarding tasks 1 and 2 (performance and usability evaluation, respectively), we focus on the gains resulting from using our tool in contrast to commercial cursor tracking systems, among which we chose CLICKTALE because of their popularity and market share leadership. We contacted one of the most renowed graphic design firms in Spain, "Pepe Gimeno - Proyecto Gráfico". Their website is rather static; it only gets updated once a year. Designers organize the site in what they call "vintages", or yearly collections containing the most remarkable works. Each vintage comprises a single page, and displays a series of screenshots (Figure 9). When the site requires updating, two webmasters from an outsourced web design studio commit the necessary changes.

The aforementioned webmasters were told to instrument the website both with SMT2ϵ and CLICKTALE systems at the same time, using default settings. For the former, they were sent the SMT2ϵ files as a ZIP archive, since our system was not publicly available at that moment. For the latter, they subscribed a free plan at http://clicktale.com.

Users were tracked approximately for two months. Eventually, 2084 visits (1738 after outlier removal) were collected with SMT2ϵ and 393 with CLICKTALE (the maximum recording amount allowed by CLICKTALE's free plan is 200 visits per month).

Fig. 9: Home page of the website tracked in this evaluation.

A log was considered an outlier when browsing time exceeded 1.5 the value of the interquartile range (IQR). This technique is very common in data mining, i.e., removing samples from analysis that do not lie between Q1 - 1.5 IQR an Q3 + 1.5 IQR. This way, we could deal with more consistent observations from the gathered logs.

5.1. Performance Evaluation

5.1.1. Performance Study 1. The first experiment aimed to compare both systems in terms of logging performance. We reproduced the exact amount of bytes that were transmitted by SMT2ϵ, by retrieving the stored data from the tracking database and converting them to the 'application/x-www-form-urlencoded' format—which is the format that both SMT2 and SMT2ϵ use to transmit the data over the network. We remark that the only difference between SMT2 and SMT2ϵ in terms of logging capabilities is that SMT2ϵ features LZW compression.

At present, CLICKTALE transmits data in chunks of 1 kB via Ajax GET requests, perhaps to keep request URIs short and thus avoid 414 HTTP error codes[8]. Thus, logged data are buffered on the client side until reaching 1 kB, in which case data are sent and the buffer is flushed. Therefore, we compared SMT2ϵ against this baseline of 1 kB, given our inability to manually modify CLICKTALE's logging scripts. The results of this study are shown in Table I.

Table I: Recording performance of the compared systems per pageview. SDs, where available, are denoted in parentheses.

System	# Logs	Browsing Time (s)	Data Size (kB)	# HTTP requests
SMT2ϵ	1738	76.84 (103.1)	2.64 (2.7)	2.56 (5.9)
CLICKTALE	393	74	1	10.58 (23.1)

In order to facilitate comparison, the number of HTTP requests submitted by CLICKTALE was estimated according to the raw data size captured by SMT2, since CLICK-

TALE does not provide such a detailed information. As we shall discuss in the next section, SMT2ϵ achieves a compression ratio of around 1:5 in comparison to SMT2, which is consistent with the 'Transmitted Data Size' results reported in Table I. In addition, as observed in the table, average browsing time reported by CLICKTALE and SMT2ϵ is similar, which gave us another hint that our estimates may be adequate. We conclude that the number of HTTP requests that CLICKTALE performs per pageview is significantly higher than that of SMT2ϵ [$t(2129) = -13.51, p < .0001$]. Further, effect size suggested moderate practical significance (Cohen's $d = 0.71$, Pearson's $r = 0.33$). On the contrary, the data size that CLICKTALE transmits is less than half the size of that transmitted by SMT2ϵ. Differences were also found to be statistically significant, with a moderate effect size [$t(2129) = 11.70, p < .0001, d = 0.67, r = 0.31$]. A discussion about these observations is provided in Section 6. We must say that SMT2ϵ features continuous recording by default, which was the setting used in this evaluation, so that tracking was enabled even when the browser window was not in focus (e.g. minimized in the background or occluded by a foreground application). Hence, if intermittent recording were set instead, perhaps much less data would have been transmitted.

5.1.2. Performance Study 2. To provide more insights about SMT2ϵ's logging performance, the following experiment aimed to compare it against its predecessor, SMT2. We instrumented the home page at http://smt.speedzinemedia.com, and visitors were randomly tracked at different frame rates, namely $10, 20, \ldots, 50$ Hz. Eventually, 1200 visits were collected, which makes 240 logs in total per sampled frequency. In order to make this study comparable to the previous one, we used continuous recording mode.

We followed the same procedure to reproduce SMT2ϵ's transmitted data size as depicted in the previous study. This time, users were tracked at most for 30 seconds, matching SMT2ϵ's default `postInterval` property. This way, in all cases, logging required exactly one HTTP request. Cursor data were stored on our server.

In Table II, the baseline case (uncompressed data) corresponds to SMT2, while compressed data corresponds to SMT2ϵ. Together with Figure 10, it can be observed that SMT2ϵ is more efficient than its predecessor for transmitting cursor data. In all cases, differences are statistically significant ($p < .0001$, two-tailed t-tests) with high effects sizes ($d > 1, r > 0.5$).

Since only one HTTP request was required to log cursor data, if compared to CLICKTALE's logging approach, CLICKTALE would have required to perform approximately 3.52 requests on average (SD=1.1) to log the same data size. This estimation is made by interpolating in Table II the uncompressed data sizes corresponding to 20 and 30 Hz, since the default recording frame rate of SMT2ϵ is 24 Hz. The reader is redirected to Section 6 for a series of comments about logging performance.

Table II: Network overhead incurred by logging at different frame rates (see also Figure 10).

Hz	Browsing Time (s)	Uncompressed Log Size (kB)	Compressed Log Size (kB)
10	19.13 (10.7)	1.96 (1.0)	0.94 (0.4)
20	13.94 (10.6)	2.72 (1.9)	1.07 (0.5)
30	15.06 (11.4)	4.33 (3.1)	1.35 (0.7)
40	14.63 (14.3)	5.38 (4.1)	1.67 (1.0)
50	16.45 (10.6)	7.71 (4.9)	2.40 (1.5)

5.2. Usability Evaluation

In this study, we return to comparing SMT2ϵ and CLICKTALE; now in terms of usability. Five participants (three graphic designers and two webmasters) that tested both

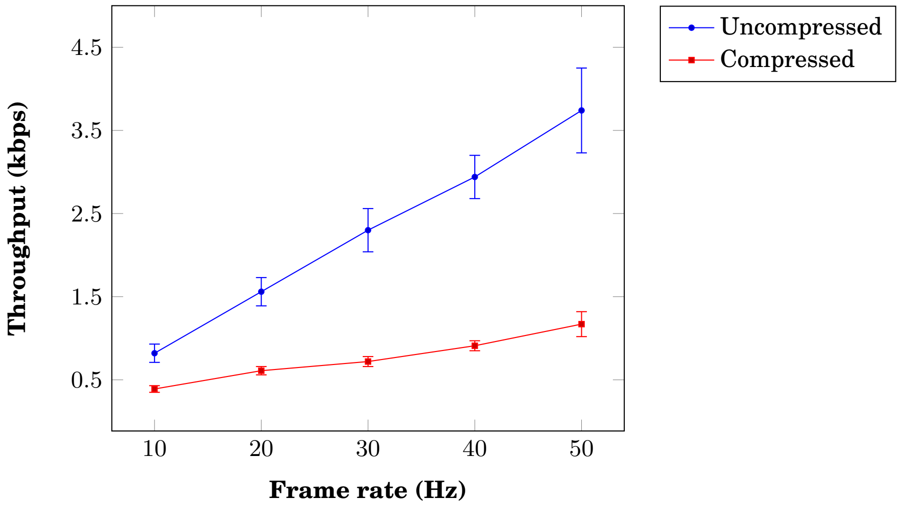


Fig. 10: Bandwith required by SMT2 (uncompressed) and SMT2ϵ (compressed) to transmit cursor data over the network. Error bars denote 95% Confidence Intervals.

SMT2ϵ and CLICKTALE at http://pepegimeno.com took part in the following studies. None was a usability expert or had previous experience in using web tracking tools (they were only familiarized with Google Analytics). Participants used both systems almost in a daily basis. They were interviewed before and after dealing with both tracking systems.

5.2.1. Case Study Overview. Overall, participants found both systems very helpful. The main advantage suggested was being able to reproduce exactly what users did on a web page. Anecdotally, participants seemed to be more technically inclined to using SMT2ϵ, mainly because of the customizations it could support and its openness. However, they complained about the lack of a more detailed documentation in SMT2ϵ. In this regard, they appreciated CLICKTALE's help desk and online support.

Regarding visualization possibilities, SMT2ϵ was perceived as being most complete, but also more difficult to start with. Some participants suggested that this could be a consequence of having integrated all visualization possibilities in a single section on the admin site (see Appendix A). As a result, they reported that CLICKTALE features a clear organization and hence the learning curve may be smoother. Nevertheless, a common complaint regarding Clicktale was that visualizations are displayed inside an iframe and scroll bars appear quite often. This was perceived as especially bothersome by the interviewees.

Everybody liked the option of being able to switch SMT2ϵ's hypervideos to a static representation by pressing a single key stroke, specially when working with a large number of aggregated logs. Moreover, watching replays of simultaneous logs was perceived as a particularly time-saving feature. In fact, they did expect CLICKTALE's replays would provide some kind of aggregation facilities, and were surprised of not being able to interact with the logged data this way.

Participants also found very helpful the possibility of adding annotations to hypervideos. An interesting comment by one participant was that he created empty hypernotes to mark those videos he found most interesting. Everyone saw the value of hyperfragments, although they reported that hyperfragments would be more aptly for large design teams to make the most of their potential. For instance, one participant mentioned that it would be helpful to share a link of a video isolating a particular browsing behavior like reading or scrolling.

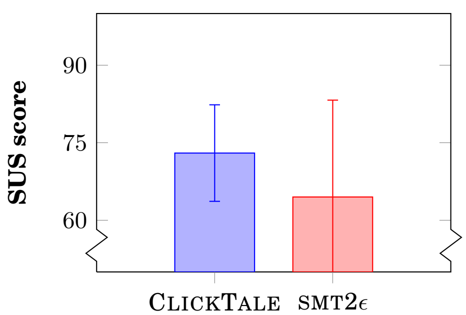
Participants observed that people typically do not click while browsing the website, beyond those clicks required to navigate to other pages. In this regard, besides not having any experience in usability or web interaction, designers felt that this happened because links are not very prominent and hence they might go unnoticed most of the time. Since pages were allocated a couple of minutes on average (see Table I), they hypothesized that lack of clicking had no association with poor content structuring.

The observations thrown by both tracking systems provided designers with a more detailed thinking about the website. They set targets for the next year that would not have been thought of if they would not have tested both systems. For instance, they decided to add more interactivity to vintages, and perhaps make images smaller to keep pages short in terms of vertical scrolling.

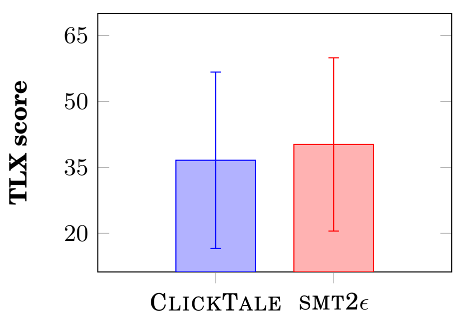
Finally, we did not get the chance of testing SMT2ϵ's automatic profile generation capabilities (behavioral clustering module) with our participants. Being this feature exclusive of our system, there is no reasonable comparison to other tools for obtaining the same information at present.

5.2.2. Measuring System Workload and User Subjectivity. Beyond the previous qualitative observations commented above, we also measured more tangible factors of both tracking systems. When the 2-month tracking campaign finished, participants filled in a couple of online questionnaires: one focused on measuring system usability and other focused on measuring participants workload. The results are shown below.

Regarding user subjectivity, we used the System Usability Scale (SUS) questionnaire [Brooke 1996]. SUS comprises a 10-item scale giving a global view of subjective assessments of usability. SUS scores range from 0 to 100 (the higher the score, the better). As observed, both systems can be considered to be similarly perceived in terms of SUS: 73 (SD=10.6) for CLICKTALE and 64.5 (SD=21.3) for SMT2ϵ. Differences are not statistically significant, with a near-moderate effect size [$t(8) = -0.79, p = .449, d = 0.56, r = 0.26$].



(a) SUS questionnaire.



(b) TLX questionnaire.

Fig. 11: Overall usability study results. Error bars denote 95% Confidence Intervals.

Regarding system workload, we used the NASA Task Load Index (TLX) [Hart 2006]. TLX uses six dimensions to assess mental workload: mental demand, physical demand, temporal demand, performance, effort, and frustration. Workload scores range from 0 to 100 (the lower the scores, the better). Overall, both systems can be considered to be similarly perceived in terms of TLX: 36.6 (SD=22.9) for CLICKTALE and 40.2

(SD=22.5) for SMT2ϵ. Differences are not statistically significant, with a small effect size [$t(8) = 0.24, p = .811, d = 0.17, r = 0.08$].

A more detailed examination of the data revealed that each system scored slightly better than the other in half of the analyzed dimensions. Differences were found to be statistically significant only in terms of mental demand yet with a small effect size [$t(8) = 2.54, p = .034, d = 0.14, r = 0.06$]. This finding was supported by the comments users made about SMT2ϵ requiring more cognitive effort than CLICKTALE to begin with. For the remaining dimensions analyzed, the t-tests reported $p > .05$ in all cases, with small to moderate effects sizes ($0.14 < d < 0.62, 0.07 < r < 0.29$), which suggests that both systems perform approximately the same way.

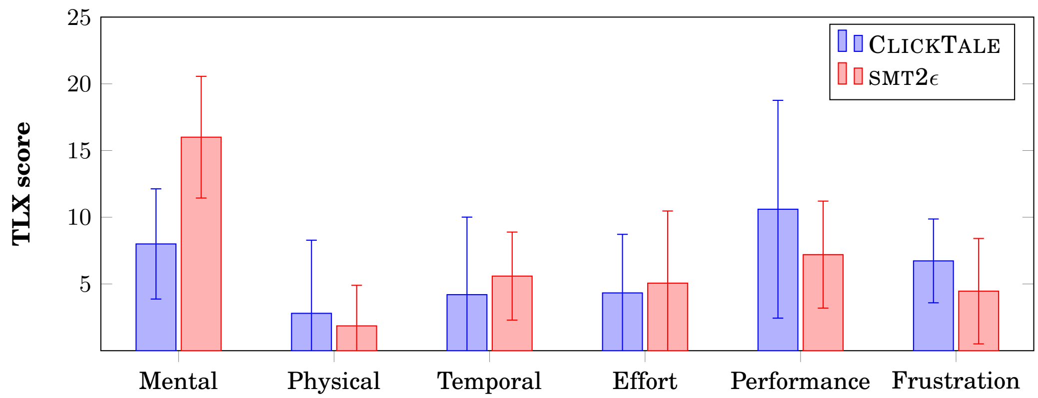


Fig. 12: Results of the six TLX dimensions. Error bars denote 95% Confidence Intervals.

5.3. Prediction Experiments

SMT2ϵ automatically extracts cursor features that can be used for user and page modeling (see Figure 8). Therefore, we felt that a study to assess the validity of these features was necessary. To illustrate it, we decided to run a series of experiments to predict time spent on page, activity (amount of mouse movements), number of clicks, and vertical scroll reach. This decision was three-fold. First, research have shown that these features are mostly correlated with page relevance and user interest [Hijikata 2004; Guo and Agichtein 2012], and may have diverse applications in real-world contexts; e.g., for website personalization, recommender systems, or web search results reranking. Second, these features are already available in the gathered logs, so they can be used as ground truth data to measure prediction accuracy, without having to manually label each log. Third, these experiments illustrate something that cannot be done with commercial systems, since commercial systems own the users' data, and such data are not accessible in raw form.

Recapitulating from Section 4.4, SMT2ϵ automatically extracts 15 cursor features from gathered logs: browsing time (1d), cursor activity (1d), number of clicks (1d), cursor distance (1d), trail length (1d), cursor range (2d), scroll reach (2d), entry point (2d), exit point (2d), and cursor centroid (2d). We denote in parentheses the number of dimensions of each feature i.e., 1d means a single value, while 2d means that the feature has an horizontal and a vertical component.

For each experiment, the feature that would be predicted was modeled as a linear combination of the remaining 14 features. Models were fitted using ordinary least squares for multiple regression. In order to decide the best features for the final fitting, we used the Information Criterion (IC) approach, which minimizes the information loss of the fit:

$$\mathrm{IC}(k) = -2\log(\mathcal{L}) + g \tag{1}$$

where $\mathcal{L}$ is the maximum likelihood for the model. When $g = 2k$, Equation 1 is the Akaike's information criterion (AIC), while $g = k\log(n)$ is the Bayesian information criterion (BIC), being k the number of parameters in the model and n the number of observations. AIC and BIC are both based on the idea that minimizing the relative entropy between the 'true' distribution and the tentative model yields the optimal model.

We chose BIC for our experiments since, according to the literature, AIC frequently prefers a more complex model over a simpler model [Burnham and Anderson 2004], and it may fail to converge in probability to the true model, whereas BIC tends to choose a simpler model and does converge as n increases [Raftery 1995].

After the fitting, we used a variant of K-fold cross validation to evaluate the accuracy of the models. Up to 50 times, we allocated two data partitions: one for training and other for testing. Observations were randomly assigned to a partition, resulting in 70% of the samples (1216 logs) for training and 30% (522 logs) for testing on each iteration. Therefore, a model derived from the training partition would be used as a binary classifier on the testing partition. A classification error was considered when the difference between the predicted feature and its true value was higher than (or equal to) the residual standard error provided by the model. Experiments were performed with the R stats package.

5.3.1. Results. Table III provides an overview of the experimental results, while the best fitting models are given in tables IV to VII.

Table III: Overview of prediction results using the models for classification. SDs are denoted in parentheses.

Predicted Feature	**% Accuracy**	**Adjusted R^2**
browsing time	84.95 (1.9)	0.49 (0.02)
no. clicks	91.79 (1.3)	0.21 (0.01)
cursor activity	74.06 (2.1)	0.24 (0.01)
scrollY	94.21 (1.3)	0.24 (0.02)

Regarding time spent on page, prediction accuracy is 84.95% (SD=1.9). All regressors in Table IV add statistical significance to the model ($p < .0001$); however, it can be observed that trail distance and trail length could be considered weak predictors, with extremely small coefficients. The adjusted R^2 is 0.49, which suggests that the model is fairly adequate to predict browsing time for unseen data. (Adjusted R^2 decreases as predictors are added to the model, and this model uses 9 out of 14 predictors.)

In addition, the model reports a negative coefficient for cursor activity, which is consistent as theory predicts, i.e., a small amount of mouse movements indicates that browsing time is likely to be high. This observation is also consistent with the model shown in Table VI. Similarly, the number of clicks is typically expected to be proportional to the time spent on page, which is also reported by the model shown in Table V, with a positive coefficient in both cases. These observations suggest that prediction of browsing time can be derived from cursor features with good prospects of success.

For the other three models, prediction accuracy was found to be equally satisfactory, specially regarding clicks (M=91.79%, SD=1.3) and vertical scroll reach (M=94.21%, SD=1.3). Similarly to previous observations, model coefficients were found to be consistent with theory. For instance, it was expected that both entry coordinate and vertical cursor range will influence in scroll reach, which is something reported by the model shown in Table VII. Nevertheless, adjusted R^2 scores for the remaining models were around 0.2, which suggests that these models may be affected by other factors beyond the interaction features provided by SMT2ϵ. As such, more research to build better models seem a promising venue for future work.

Table IV: Regressor coefficients to predict time spent on page, according to the model that provided the best classification accuracy (89.61%). All features were found to be statistically significant ($p < .0001$) at the $.05$ level.

Feature	**Coefficient**	**Std. Error**
(intercept)	10.7	3.3
cursor activity	-1.1	$1.4{\cdot}10^{-1}$
no. clicks	1.2	$2.1{\cdot}10^{-1}$
trail distance	$-1.3{\cdot}10^{-4}$	$2.1{\cdot}10^{-5}$
trail length	$4.2{\cdot}10^{-8}$	$1.6{\cdot}10^{-8}$
entryX	$4.1{\cdot}10^{-2}$	$8.8{\cdot}10^{-3}$
exitY	$3.7{\cdot}10^{-3}$	$7.4{\cdot}10^{-4}$
rangeX	$6.3{\cdot}10^{-2}$	$8.2{\cdot}10^{-3}$
rangeY	$8.7{\cdot}10^{-3}$	$7.2{\cdot}10^{-4}$
centroidX	$-5.5{\cdot}10^{-2}$	$1.2{\cdot}10^{-2}$

Table V: Regressor coefficients to predict number of clicks, according to the model that provided the best classification accuracy (95.19%). All features were found to be statistically significant ($p < .0001$) at the $.05$ level.

Feature	**Coefficient**	**Std. Error**
(intercept)	$-5.5{\cdot}10^{-1}$	$5.2{\cdot}10^{-1}$
browsing time	$5.0{\cdot}10^{-2}$	$3.8{\cdot}10^{-3}$
rangeX	$-4.0{\cdot}10^{-3}$	$9.8{\cdot}10^{-5}$
scrollX	$9.0{\cdot}10^{-2}$	$1.4{\cdot}10^{-2}$

Table VI: Regressor coefficients to predict cursor activity, according to the model that provided the best classification accuracy (78.07%). All features were found to be statistically significant ($p < .0001$) at the $.05$ level.

Feature	**Coefficient**	**Std. Error**
(intercept)	8.28	$8.3{\cdot}10^{-1}$
browsing time	$-7.8{\cdot}10^{-2}$	$6.8{\cdot}10^{-3}$
entryX	$1.2{\cdot}10^{-2}$	$1.5{\cdot}10^{-3}$
exitY	$-4.1{\cdot}10^{-4}$	$1.3{\cdot}10^{-4}$
rangeX	$1.5{\cdot}10^{-2}$	$1.5{\cdot}10^{-3}$
rangeY	$4.0{\cdot}10^{-4}$	$1.2{\cdot}10^{-4}$
centroidX	$-6.3{\cdot}10^{-1}$	$2.2{\cdot}10^{-3}$

Table VII: Regressor coefficients to predict vertical scroll reach, according to the model that provided the best classification accuracy (96.53%). All features were found to be statistically significant ($p < .0001$) at the $.05$ level.

Feature	**Coefficient**	**Std. Error**
(intercept)	11.49	13.1
entryY	$6.0 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$
rangeY	$3.8 \cdot 10^{-2}$	$2.0 \cdot 10^{-3}$
scrollX	$6.8 \cdot 10^{-1}$	$2.4 \cdot 10^{-1}$
centroidX	$-6.5 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$
centroidY	$-6.1 \cdot 10^{-2}$	$4.2 \cdot 10^{-3}$

6. GENERAL DISCUSSION

Clearly, there is a compromise solution between tracking frequency and visualization coarseness, i.e., the higher the frame rate, the better the replay quality, but also the higher the amount of data to capture, and therefore the higher the overhead incurred by logging. Ensuring a low overhead is specially important in mobile contexts, where battery life is limited and where networks can be slow and very expensive. Therefore, using frame rates around 20 fps seem to be a well-balanced recommendation. Even better, another possibility would be recording user interaction only when a browser event is fired. This would require some modifications to SMT2ϵ in order to synchronize different cursor streams, but it would contribute nevertheless to reduce (even more) logged data size.

On the other hand, as we have shown in Section 5.1, SMT2ϵ has a very small footprint in terms of number of HTTP requests performed. This also contributes to reduce overheads incurred by logging, as HTTP requests introduce an important load increase on the server. Not only there is a header block for both the request and the response, but there is also a three-way handshake due to the TCP/IP layer, that underlines the transmission protocol. So, in the absence of web server configuration tweaks, it is commonly agreed that transmitting more data in less HTTP requests performs better than the opposite case. This is another opportunity for industry systems to improve logging performance.

Concerning cursor data compression, we must mention that SMT2ϵ features LZW encoding for practical reasons. One might think that Huffman coding would be particularly appropriate instead. However, using Huffman codes to compress cursor data is not optimal. The space saved while compressing the data is then penalized with the size of the Huffman tree, which must be transmitted to the server in order to reconstruct (decode) the compressed data. If the tree were always known and fixed, then Huffman coding could outperform LZW. Nevertheless, each cursor trail is unique by definition, having thus a unique decoding tree, and therefore the data must be transmitted together with the tree. So, the size of a Huffman-compressed cursor trajectory plus its corresponding decoding tree is typically higher than the size of the data encoded with LZW. Perhaps some hybrid strategy would be worth trying in a future; e.g., using Huffman to compress cursor data and LZW to compress Huffman trees.

Regarding usability evaluation, we are really happy with the results. With around 20 employees[9], CLICKTALE is a leader in the cursor tracking industry, while SMT2ϵ is a single-person's work. Overall, we only observed statistically significant differences regarding mental workload in the TLX test, where SMT2ϵ scored higher than CLICKTALE, suggesting that SMT2ϵ is more mentally demanding to operate. However, for the 5 remaining dimensions as well as the SUS scores no statistical significance was found.

On another level, we have shown a series of cursor features that can be predicted with high accuracy on average (78% in the worst case, when predicting cursor activity). Nevertheless, we capitalize on the fact that the purpose of the prediction experiments is merely illustrating that SMT2ϵ provides raw cursor data that are both accessible and easily manipulable, and can be used to test theories on user behavior using real observations. Other features could also provide more insights about page-level interactions. For instance, being able to predict the centroid coordinate would allow the webmaster to consider placing interesting content near that coordinate, or a link to some promoted page. In addition, whether a model is really a good fit or not depends on the context of the study. Using a more advanced technique for multiple regression may be desirable to increase precision and fit quality, although it is definitely outside the scope of this paper.

To conclude, we remark that most of these studies would have never been possible without a system like SMT2ϵ, since current commercial tools do not allow users to manipulate tracking data in such a fine-grained detail.

7. LIMITATIONS

Web-based activity tracking systems have inherent limitations, and of course SMT2ϵ is no exception to this rule. Although measuring page-level interactions is cheaper and enables remote data collecting at large, the main drawback we have found is that assessing the allocation of visual attention based on interaction data alone is a non-trivial task. For instance, while it is commonly agreed that "a mouse cursor can tell us more" [Chen et al. 2001], other researchers [Huang et al. 2012] have demonstrated that browsing time and user behavior have notable repercussions on gaze and mouse cursor alignment. Also, it has been shown that users do not necessarily attend to what they are looking at, and they do not necessarily look at what they are attending to [Toet 2006]. Therefore, the usability practitioner should be aware of these facts before considering using a web tracking system, depending on the task that would be assessed or the context of their study.

On the other hand, our tool was designed to handle a limited number of simultaneous user sessions in the same hypervideo. One may note that if the system were used to show data from, say, 10000 concurrent users, then we believe the video visualization would not be much meaningful. Suffice to say it could be done, but at the cost of increasing the cognitive overload for the viewer (since visually inspecting too many users at the same time can be stressful), and only limited by the processing power of his computer. In this situation, aggregated data would work much better if rendered as a single image—discarding thus the temporal information but retaining interactivity for the viewer. This way, it is still possible to visually infer time-based properties such as mouse velocities (e.g., by looking at the 'directions & distances' layer, Figure 7).

Additionally, besides the fact that our tool normalizes the mouse coordinates to avoid possible visual biases while replaying the hypervideos, we have noticed that sometimes visualization is not perfectly accurate, partly due to JavaScript rounding errors, partly due to discrepancies between how browsers render CSS. As such, in these cases a few pixels can make a difference: image for instance a Firefox user that clicked on the very top-right corner of an image link; if the video were rendered on the same screen dimensions but on a different browser, then the viewer could see that user clicking on an empty space (depending of course on which CSS rules where established). These browser discrepancies can be greatly minimized by using a reset stylesheet on the web page. On the contrary, higher discrepancies are expected when the user access from a mobile device and the viewer uses a desktop computer. We are currently investigating different methods that would tackle this problem, which is common to all web-based tracking systems, and for which there is no direct solution. For instance, the system could use the mobile user agent to fetch the page the user had visited, but it could happen that the page had been updated, or even that it no longer exists. The same argument may apply to the stylesheets of that page. Therefore, a technically more advanced approach should be taken into consideration, such as caching all the assets for each single user session, at the cost of increasing the storage space on the tracking database.

Related to the previous issue, we should mention that, depending on page complexity, it could be the case that the screen shown in the playback is not *exactly* the one the users saw. For instance, JavaScript prompts, pulldown menus, or Ajax calls that validate form data or modify the DOM at runtime would not be shown during the replay. This can be solved by triggering events that relate to the cursor trajectory (e.g., hovering, clicking, etc.) or other registered browser events (e.g., keypresses). However, this approach could have undesired side effects, especially when replaying multiple logs simultaneously. For instance, event triggering may cause navigating away from the current page, inserting redundant information into the website database, or resubmitting a shopping cart form. As a result, we decided not to implement event triggering for purely practical reasons.

Finally, it is worth pointing out that some users may utilize other I/O devices to assist web browsing, such as screen readers or speech recognizers. We acknowledge that in this case no cursor tracking system would be appropriate to study the behavior of those users, and in fact we encourage the expert practitioner to consider other input signals, if available, when running usability tests. For instance, if an eye-tracker were used, it would be possible to easily synchronize both coordinate streams, as indicated, e.g., by Huang et al. [2011; 2012].

8. CONCLUSION

To better understand web-based interaction, current tracking systems should rely on the browsing capabilities of the users, instead of traditional server access logs. We believe delivering interaction data as a hypervideo for assessing the usability of websites is a promising and helpful idea.

This paper has described the design and implementation of SMT2ϵ, a tool for automatically gathering, mining, and visualizing browsing data in an interactive hypermedia presentation. SMT2ϵ collects fine-grained information about user behavior, and allows viewers to control what they watch, when, and *how*, by selecting diverse types of infographics. Our proposal is part of a larger move to bring more interactivity into browsing data exploration and analysis.

We have reported the main differences between SMT2ϵ and both past and current web tracking systems, including its predecessor (SMT2). We also have shown the value of enhancing video visualizations with interactive techniques to present the viewer with complex information quickly and clearly. A series of experimental evaluations have illustrated that ours is an efficient implementation, and that it performs sometimes better than a leader commercial system.

Tracking page-level browsing activity with SMT2ϵ requires no real effort from the user, other than standard usage. It also requires no training and provides context for actions. Armed with this awareness, one may conduct qualitative or quantitative studies to complement existing methodologies on web browsing behavior. We believe SMT2ϵ is ready to extend its scope to a broader, interdisciplinary audience.

9. FUTURE WORK

Since 2009, we and others have used our system for a series of research studies on different websites. At present, the tool has been downloaded more than 100K times and some of its components are being used in other projects and even in commercial products. This encourages us to keep up performing enhancements.

As previously commented in Section 7, one of our priorities for future work is implementing a more advanced caching method to minimize the rendering discrepancies; for instance, when the same URL is accessed from mobile devices and desktop computers. We also plan to work actively on scalability and performance limits, especially concerning high-recording speeds. In this regard, we are currently testing the HTML5 WebSocket API for transmitting data. Using this technology would also enable real-time activity sharing, were users and viewers interact concurrently (e.g., think of an online customer support service). Another possibility is investigating new types of cursor data compression, so that the logging footprint can be further minimized.

Another line of research would be leaned towards enriching the system with other types of behavior analysis, such as working with eye-tracking data, since user interaction is inherently multimodal. Therefore, considering more independent data sources can only make evaluation studies stronger. Finally, the reader is encouraged to try SMT2ϵ, which is open source software that can be inspected at no cost and downloaded from http://smt2.googlecode.com.

Notes

1 http://httpd.apache.org/docs/current/logs.html
2 http://clicktale.com
3 http://userfly.com
4 http://mouseflow.com
5 http://m-pathy.com
6 http://clixpy.com
7 http://otterplus.com/mps
8 http://www.faqs.org/rfcs/rfc2616.html
9 http://www.aboutanalytics.com/clicktale

APPENDIX

Monitoring user interactions at the page-level can be very useful to help shaping a more usable website, or making it more appropriate to the skills and abilities of their users. However, as in other web tracking applications, this work raises privacy concerns. We are interested in understanding web browsing behavior, but we also want the user to be respected, so we designed the SMT2ϵ system with that notion in mind.

First, we believe logging keystrokes could be employed for unfair purposes, depending on the uses that one could derive from this tool. For that reason, we rejected to log raw keystroke data and track only keyboard events instead, without registering the associated character codes. Second, we believe users should not be monitored without their consent. This is a webmaster's responsibility, but not doing so could be considered unethical in some countries. Therefore we recommend to ask always the user before tracking takes place. Furthermore, once a user has agreed to track, we advocate for asking her consent again after a prudential amount of time (e.g., a few hours, until the end of the browsing session, or when a tracking campaign finalizes). Third, we believe logged data should be stored in a server the webmaster owns, and not in one she cannot control. At least, it should be possible to let users access their (raw) data. We encourage commercial tracking systems to do so, since chances are there and current web technologies can support it. Finally, unlike most analytics packages that track other sites users have visited or the searches they have made, we do not collect other information than basic browser events derived from normal usage at the site where SMT2ϵ is included. This way, we try to avoid an illegitimate abuse of our system (e.g., without advising at all that users are being tracked or hijacking submitted form data). Above all, the ethical use of computers should be above any functionality or feature.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The authors wish to thank TWEB reviewers for their helpful comments and suggestions to improve this manuscript. The authors also thank the staff members at pepegimeno.com and nectarestudio.com who volunteered to took part in our evaluation studies.

REFERENCES

AGICHTEIN, E., BRILL, E., AND DUMAIS, S. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th conference on Research and development in information retrieval (SIGIR)*. 19–26.

ARROYO, E., SELKER, T., AND WEI, W. 2006. Usability tool for analysis of web designs using mouse tracks. In *Proceedings of the extended abstracts on Human factors in computing systems (CHI EA)*. 484–489.

ATTERER, R., WNUK, M., AND SCHMIDT, A. 2006. Knowing the user's every move – user activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the WWW conference*. 203–212.

BROOKE, J. 1996. SUS: A "quick and dirty" usability scale. In *Usability Evaluation in Industry*, P. Jordan, B. Thomas, B. Weerdmeester, and A. McClelland, Eds. Taylor and Francis.

BURNHAM, K. P. AND ANDERSON, D. R. 2004. Multimodel inference: Understanding aic and bic in model selection. *Sociological Methods Research 33,* 2, 261–304.

BUSCHER, G., WHITE, R. W., DUMAIS, S., AND HUANG, J. 2012. Large-scale analysis of individual and task differences in search result page examination strategies. In *Proceedings of the fifth ACM international conference on Web search and data mining (WSDM)*. 373–382.

CHEN, M. C., ANDERSON, J. R., AND SOHN, M. H. 2001. What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. In *Proceedings of the extended abstracts on Human factors in computing systems (CHI EA)*. 281–282.

DANIEL, G. AND CHEN, M. 2003. Video visualization. In *Proceedings of the IEEE Conference on Visualization (VIS)*. 409–416.

FRANCISCO-REVILLA, L. 1998. A picture of hypervideo today. Tech. rep., Center for the Study of Digital Libraries: Texas A&M University.

GUO, Q. AND AGICHTEIN, E. 2010a. Ready to buy or just browsing? detecting web searcher goals from interaction data. In *Proceedings of SIGIR*. 130–137.

GUO, Q. AND AGICHTEIN, E. 2010b. Towards predicting web searcher gaze position from mouse movements. In *Proceedings of the extended abstracts on Human factors in computing systems (CHI EA)*. 3601–3606.

GUO, Q. AND AGICHTEIN, E. 2012. Beyond dwell time: Estimating document relevance from cursor movements and other post-click searcher behavior. In *Proceedings of the 21st international conference on World Wide Web (WWW)*. 569–578.

GUYON, I., SCHOMAKER, L., PLAMONDON, R., LIBERMAN, M., AND JANET, S. 1994. UNIPEN project of on-line data exchange and recognizer benchmarks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*. 29–33.

HART, S. G. 2006. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the 50th Annual Meeting on Human Factors and Ergonomics Society (HFES)*. 904–908.

HAUGER, D., PARAMYTHIS, A., AND WEIBELZAHL, S. 2011. Using browser interaction data to determine page reading behavior. In *Proceedings of the 19th international conference on User modeling, adaption, and personalization (UMAP)*. 147–158.

HAUGER, D. AND VAN VELSEN, L. 2009. Analyzing client-side interactions to determine reading behavior. In *Proceedings of Adaptivity and User Modeling in Interactive Systems (ABIS)*. 11–16.

HIJIKATA, Y. 2004. Implicit user profiling for on demand relevance feedback. In *Proceedings of the 9th international conference on Intelligent user interfaces (IUI)*. 198–205.

HIRATA, K., HARA, Y., SHIBATA, N., AND HIRABAYASHI, F. 1993. Media-based navigation for hypermedia systems. In *Proceedings of the fifth ACM conference on Hypertext (HT)*. 159–173.

HUANG, J., WHITE, R. W., AND BUSCHER, G. 2012. User see, user point: Gaze and cursor alignment in web search. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*.

HUANG, J., WHITE, R. W., AND DUMAIS, S. 2011. No clicks, no problem: Using cursor movements to understand and improve search. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*. 1225–1234.

JACOBSEN, N. E., HERTZUM, M., AND JOHN, B. E. 1998. The evaluator effect in usability tests. In *CHI 98 conference summary on Human factors in computing systems*. 255–256.

LEIVA, L. A. 2011. Mining the browsing context: Discovering interaction profiles via behavioral clustering. In *Adjunct Proceedings of the 19th conference on User Modeling, Adaptation, and Personalization (UMAP)*. 31–33.

LEIVA, L. A. AND VIDAL, E. 2010. Assessing user's interactions for clustering web documents: a pragmatic approach. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia (HT)*. 277–278.

LEIVA, L. A. AND VIVÓ, R. 2012. Interactive hypervideo visualization for browsing behavior analysis. In *Proceedings of the 21st international conference companion on World Wide Web (WWW)*. 381–384.

LIESTØL, G. 1994. Aesthetic and rhetorical aspects of linking video in hypermedia. In *Proceedings of the 1994 ACM European conference on Hypermedia technology (ECHT)*. 217–223.

MUELLER, F. AND LOCKERD, A. 2001. Cheese: Tracking mouse movement activity on websites, a tool for user modeling. In *Proceedings of the extended abstracts on Human factors in computing systems (CHI EA)*. 279–280.

MÜLLER-TOMFELDE, C. 2007. Dwell-based pointing in applications of human computer interaction. In *Proceedings of INTERACT*. 560–573.

RAFTERY, A. E. 1995. Bayesian model selection in social research. *Sociological Methodology 25,* 1, 111–163.

REEDER, R. W., PIROLLI, P., AND CARD, S. K. 2001. WebEyeMapper and WebLogger: tools for analyzing eye tracking data collected in web-use studies. In *Proceedings of Extended abstracts on Human factors in computing systems (CHI EA)*. 19–20.

RODDEN, K., FU, X., AULA, A., AND SPIRO, I. 2008. Eye-mouse coordination patterns on web search results pages. In *Proceedings of the extended abstracts on Human factors in computing systems (CHI EA)*. 2997–3002.

SHIFFER, M. J. 1995. Interactive multimedia planning support: moving from stand-alone systems to the world wide web. *Environment and Planning B: Planning and Design 22,* 6, 649–664.

SMITH, J. AND STOTTS, D. 2002. An extensible object tracking architecture for hyperlinking in real-time and stored video streams. Tech. Rep. 02-017, Univ. North Caroline and Chapel Hill.

TARASEWICH, P., POMPLUN, M., FILLION, S., AND BROBERG, D. 2005. The enhanced restricted focus viewer. *International Journal of Human Computer Interaction 19,* 1, 35–54.

TOET, A. 2006. Gaze directed displays as an enabling technology for attention aware systems. *Computers in Human Behavior 22,* 4, 615–647.

ŠPAKOV, O. AND MINIOTAS, D. 2007. Visualization of eye gaze data using heat maps. *Electronics and electrical engineering 74,* 2, 55–58.

Received Month Year; revised Month Year; accepted Month Year

Online Appendix to: Web Browsing Behavior Analysis and Interactive Hypervideo

LUIS A. LEIVA and ROBERTO VIVÓ, Universitat Politècnica de València

A. ADDITIONAL FIGURES

An illustrative video of SMT2 is available at http://vimeo.com/luileito/smt2-www. A video of SMT2ϵ will be available soon at http://vimeo.com/luileito/smt2e.

Figure 13 and Figure 14 provide a brief overview of the main admin sections.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

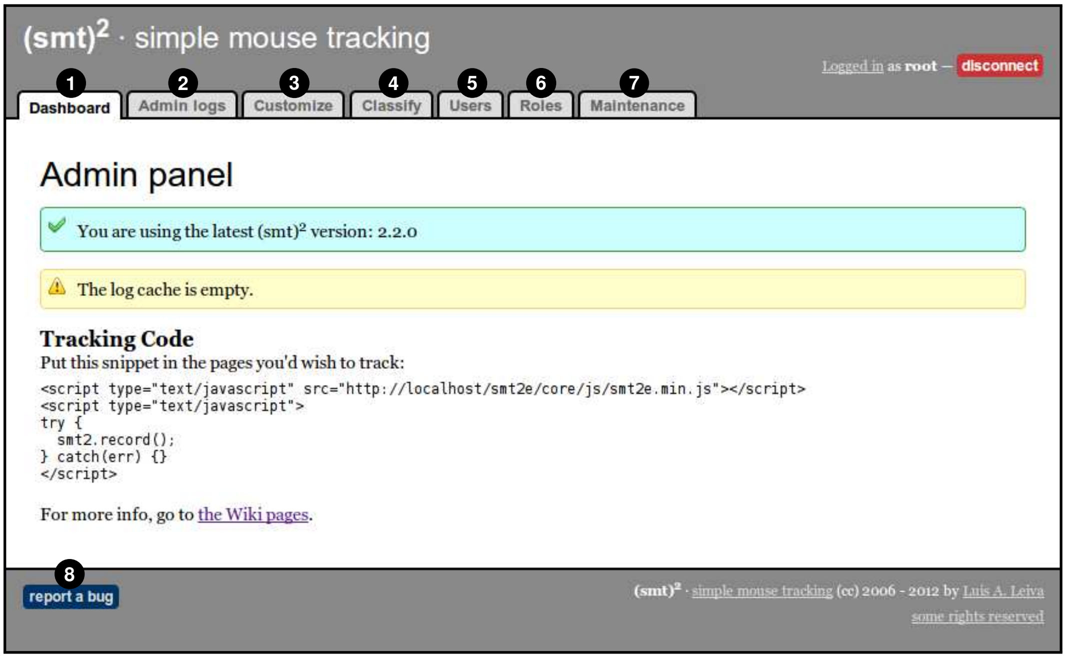


Fig. 13: Screen shown after a clean installation in a localhost setting. The dashboard ❶ provides the viewer with information about SMT2ϵ updates or new releases, server status, etc. Besides the main section devoted to manage tracking logs ❷, the admin site allows to customize preferences and general settings ❸, classify pages according to browsing behaviors ❹, create or modify user accounts ❺ and assign permissions to them ❻, and perform some maintenance tasks on the site ❼ such as backing up database tables. Finally, users can submit feedback and report issues about the system ❽ at any time.

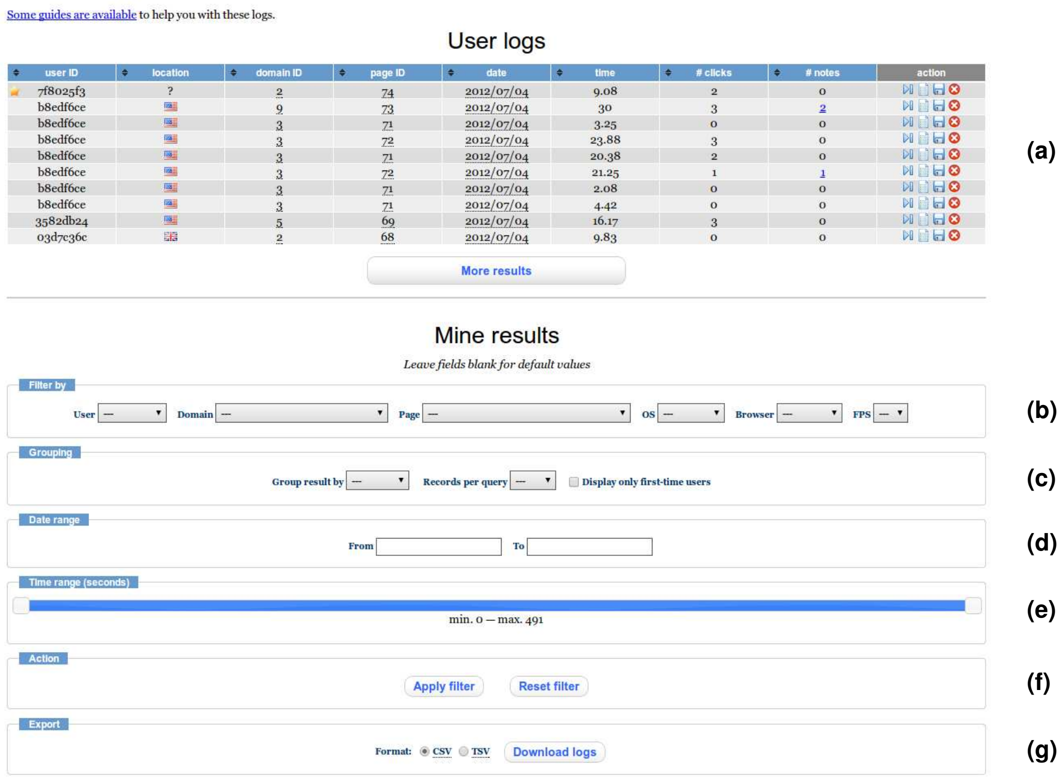


Fig. 14: The main admin section is the 'admin logs' page, where the viewer can manage the gathered data in a variety of ways. User logs are listed in the top table (14a). Possible actions to perform are visualize, analyze, download, and delete. Filtering (14b) can be done by page, user, operating system, and browser. First-time users can also be segmented, so that one can figure out how users behave when browsing a page they have not seen before. The viewer can select how many logs are going to be retrieved in each database query, and merge them by page, user, or location (14c). It is also possible to specify a date (14d) and/or time range (14e) as filtering options. At this point, the viewer can perform three actions: apply filtering (14f), reset filtering, or export the logs (14g) that match such filtering options. Downloading is supported as plain text files, so that one can easily manipulate the raw data with third party software.