IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 12, 2019, accepted April 21, 2019, date of publication May 15, 2019, date of current version June 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916996

Latent: A Flexible Data Collection Tool to Research Human Behavior in the Context of Web Navigation

CATIA CEPEDA[1,2], RICARDO TONET[1], DANIEL NORONHA OSORIO[1], HUGO P. SILVA[3,4,5], (Senior Member, IEEE), EDOUARD BATTEGAY[2,6], MARCUS CHEETHAM[2,6], AND HUGO GAMBOA[1], (Senior Member, IEEE)

[1]LIBPhys (Laboratory for Instrumentation, Biomedical Engineering and Radiation Physics), Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[2]Department of Internal Medicine, University Hospital Zurich, 8091 Zurich, Switzerland
[3]PLUX - Wireless Biosignals, S.A., 1050-059 Lisbon, Portugal
[4]IT - Instituto de Telecomunicações, 1049-001 Lisbon, Portugal
[5]EST/IPS - Escola Superior de Tecnologia do Instituto Politécnico de Setúbal, 2910-761 Setúbal, Portugal
[6]University Research Priority Program "Dynamics of Healthy Aging", University of Zurich, 8006 Zurich, Switzerland

Corresponding author: Catia Cepeda (c.cepeda@campus.fct.unl.pt)

This research was funded by the Department of Internal Medicine, University Hospital Zurich, Zurich, Switzerland and by the Portuguese Fundação para a Ciência e Tecnologia (FCT) grant I&D 2015-2020 "iNOVA4Health - Programme in Translacional Medicine" (UID/Multi/04462/2013).

ABSTRACT Internet usage has grown dramatically since the early years of its inception. The rich field of data provided by internet users in interaction with digital media content can provide insight into web-based navigation behavior and underlying psychological dimensions. Human–computer interaction in the web is an underutilized source of data for understanding human online behavior. While researchers and usability testing services do use these sources to analyze human behavior and user experience, access to the diverse range of other potentially useful data available during web-based interaction for research is limited. In this paper, we propose a novel tool in the form of a web browser extension, referred to as Latent, which can be used to simultaneously capture information from different sources while users interact with digital content. The data acquisition capabilities of Latent makes it suitable for various research purposes, ranging from studies of usability to decision-making and personality. A particular advantage of Latent is that the method and control of data acquisition is completely transparent to the user. We present the architecture of the web browser extension, describe the data that can be acquired, and report on the residual impact of the tool on the user's computer processing resources.

INDEX TERMS Browser extension, data acquisition, Human–computer interaction, web search.

I. INTRODUCTION

Since Sir Tim Berners-Lee's invention of the World Wide Web in 1989, internet usage has increased from a few thousand users to more than four billion [1]. The number of searches per day has reached 2 trillion in Google alone [2], the content has become more complex, and websites and applications have become more sophisticated. Today, it is normal for companies, products and services to have a digital website or a webstore presence and the Internet provides an essential platform for social gathering and sharing of content and experiences.

The associate editor coordinating the review of this manuscript and approving it for publication was You Yang.

The surge of e-commerce platforms, social media, websites and online services has been accompanied by extensive efforts to enhance user-friendly experiences, retain customers and users, inform customization, and conduct targeted marketing. Several tools and services allow online website tracking in order to follow users' behaviour, preferences and habits and analyse user interactions with websites and stores. Generally, these tools collect information on traffic data, most visited pages, user time per page, activity heat maps and other user-relevant data. This is done using a script embedded in the webpage content that sends data to the web service's server. The website owner then accesses a dashboard to view user interaction statistics. For research purposes, one possible

drawback with existing approaches is that they only provide a limited set of functionalities, restricted to a limited number of user interaction events, such as the number of hovers or clicks in a specific area of interest. Due to the use of specific scripts for each website, online browsing tracking is limited to supported websites, sometimes without the user's knowledge. Furthermore, it is often not possible to evaluate user interactions during sessions in which the web behaviour moves across several websites.

Some research has analysed human behaviour based on web browsing data [3]. Typically, previous work is based on features such as the duration of the visit or clickstream to model the consumer search [4], or researchers compare behaviours within and across websites [5]. Some researchers take a more complex approach to evaluate users' browsing behaviour, such as the application of text classification [6] or an algorithm to calculate the weight of a particular web page based on related actions [7]. Several scientific studies have focussed on the extraction of personality traits from keyboard and mouse behaviour [8]–[11], while others have detected and described repeated mouse movement patterns [12]. The necessary raw data for these kinds of analyses are not normally available on most tracking tools and services, which depending on the goals, are more dedicated to analysis of e-metric values or specific behaviours. Opportunities for research on users and human browsing behaviour are often hindered by the lack of flexibility of existing tools that do not therefore enable the researcher to tap into and acquire the relevant data.

This paper presents a new data collection tool to monitor web browsing behaviour, referred to as Latent. We focus in the present paper on the tool's flexibility and its impact on system performance. Latent consists of a web browser extension that is able to monitor and record all browsing behaviour. The tool does not process the data and data analysis resides with the user, thus allowing for a high degree of flexibility.

Taking into consideration that sensitive data are present in many web search activities, features are incorporated into our tool to safeguard privacy and security. For example, the extension provides the user with the means to control what parameters are recorded, allowing greater transparency. This tool has a wide range of potential applications, such as studies of user interfaces, website-specific navigation behaviour, decision-making processes or personality traits.

In the following sections, we present, firstly, related work about web browsing data acquisition tools. Second, we introduce the proposed tool in detail, focusing on the architecture data that can be acquired. Third, a validation is performed by studying the impact of the system on the computer resources. Finally, we summarize the possible information that can be extracted with the tool and present possible use case scenarios.

II. RELATED WORK

The interaction of the user during regular computer use can be recorded by programs like ViewletBuilder [13] from Qarbon, Camtasia from TechSmith, and CamCorder [14] from Microsoft Office. These systems record a video that could be edited in the end.

A wide variety of tools were developed to improve user experience. Web developers and web designers use these tools to test their websites and evaluate them according to the reports and visualization tools usually produced. Most of these software include the acquisition of mouse movement tracking and clicks, and record the session to further replay [15]–[18]. Some tools include specific reports, related to psychological analysis. CrazyEgg [15] and MouseFlow [19], for example, relate time with attention, while ClickTale [20] evaluates optimal mindset or mindstate changes. Although several tools already exist to analyse user behaviour, these are paid services used for limited purposes.

Some extensions can also be installed to monitor user activity on the web. HYFY Recorder [21] is an extension that records a video with the screen, face and voice. Remouse [22] is an extension capable of replaying the mouse movements, but without returning the original values.

Gamboa [23] developed a client-server application for web pages monitoring called WIDAM. This system has a client application that sends a message to a server for each input of the mouse or keyboard event. The message includes information about the event, the mouse position, the DOM-Object identification and time. In contrast with previous services described, this has the advantage of recording raw mouse movements data easily used by others in different studies.

A detailed comparison between the existing tools and Latent is presented in Table 1.

III. OVERVIEW OF LATENT

Latent is a data acquisition platform which gathers web browser interaction data. The architecture consists of three layers: web browser extension, web server and database (Fig. 1).

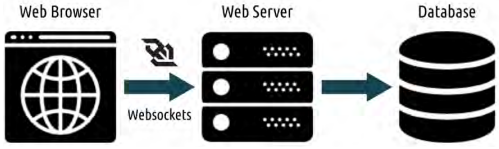

FIGURE 1. Latent platform architecture. The browser extension communicates with the web server via web sockets, which then saves the data to a database.

The main component of the platform is the web browser extension, which handles all the tasks related to data acquisition. The current version consists of a Google Chrome browser extension, although the concept works for different browsers.

The web server layer consists of a Python script that uses Tornado as the web server library [24]. Lastly, the database layer is provided by a NoSQL MongoDB database. These two layers can be replaced by custom developed layers according to the researchers' needs. The communication between a

TABLE 1. Comparison table of related work.

System	Usage Setting	Goal	Collected Data	Final User	Tracking Method	Available UI
Latent	Research	Web data collection	Mouse, keyboard, geolocation, screenshots, audio, snapshots, audio, snapshots, DOM, browser/tab settings	Researcher	Browser extension	Configuration, data export
ViewletBuilder	Research, business	Screen recorder	Video, audio	General user	Native application	Configuration, video editor
Camtasia	Research, business	Screen recorder	Video	General user	Native application	Configuration, video editor
CrazyEgg	Business	Web usage analytic	Mouse, screenshots	Web site owner	Embedded code	Data report
Inspectlet	Business	Web usage analytic, behaviour analysis	Mouse, keyboard, screenshots	Web site owner	Embedded code	Data report
Smartlook	Business	Web usage analytic, behaviour analysis	Mouse, keyboard, touch events	Web site owner	Embedded code	Data report
HotJar	Business	Web usage analytic, behaviour analysis	Mouse, keyboard, touch events	Web site owner	Embedded code	Data report
MouseFlow	Business	Web usage analytic, behaviour analysis	Mouse, keyboard, geolocation, screenshots, device type, touch events, browser/system settings	Web site owner	Embedded code	Data report
ClickTale	Business	Web usage analytic, behaviour analysis	Mouse, keyboard	Web site owner	Embedded code	Data report
HYFY Recorder	Research, business	Web usage screen recording	Mouse, screenshots, audio, snapshots	General user	Browser extension	Configuration, editor, data export
Remouse	Research, business	Mouse recording	Mouse, keyboard	General user	Native application	Configuration, data export
WIDAM	Research	Mouse recording	Mouse, keyboard	Researcher	Embedded code	Data export

browser extension and the web server is made via secure WebSockets [25].

A. WEB BROWSER EXTENSION

The web browser extension layer is organized into three components that communicate via a messaging system. These components, which depend on the Google Chrome extension architecture, are the background, the foreground and the pop-up (Fig. 2).

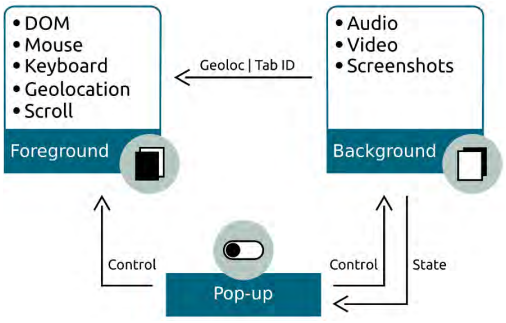

FIGURE 2. Latent web browser extension components. The components communicate via a messaging system. The pop-up exerts control on the background and foreground components. The background is responsible for acquiring media data, and the foreground for acquiring browser interaction data.

The background is always running and is responsible for the extension main logic. It has access to all browser extension APIs. It is responsible for managing the extension state,

FIGURE 3. Latent web browser extension acquisition control UI. On top, from left to right, the buttons are: Open options page; turn on/off the acquisition preview; open the acquisition results page; "about" information. The bottom button is to start/stop the acquisitions.

to acquire media devices dependent data, and also to access geolocation data. The foreground runs within each browser window tab and is responsible for acquiring navigation specific data. The pop-up consists on the user interface (UI) for acquisition control (Fig. 3). The extension also has an options page to configure the acquisition methods and media devices used (Fig. 4).

B. DATA ACQUISITION

The goal of Latent is to be as flexible as possible, allowing the highest number of possible use cases to be derived from the data collected. For that reason, we chose to enable the collection of all the available data we could collect from a web browser interaction experience. The Latent extension enables the collection of the following data: mouse interaction; keyboard interaction; geolocation; browser tab screenshots; audio from microphone; video camera snapshots; Document Object Model (DOM); and browser and

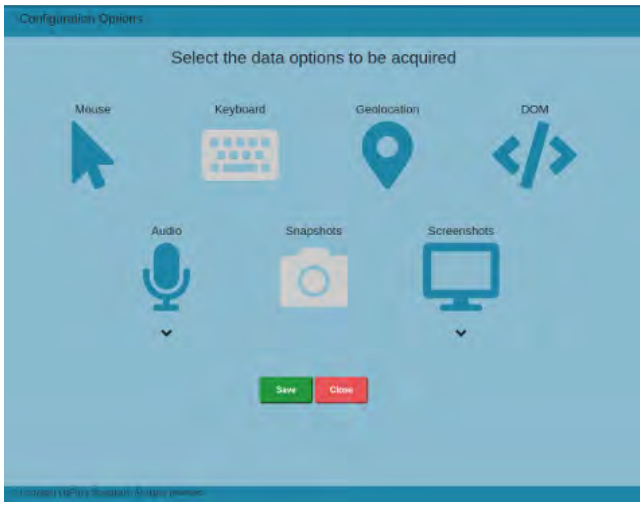

FIGURE 4. Latent web browser extension options page UI. Each icon represents an acquisition data type. When selected for acquisition they are blue, and when not selected they are grey. The Audio, Snapshots and Screenshots have extra options symbolized by the downward arrows.

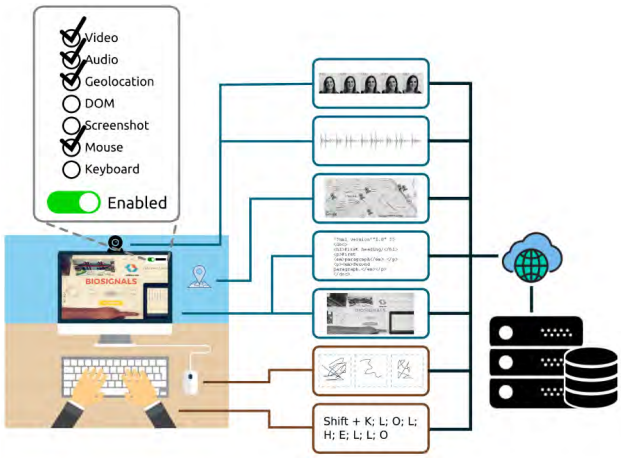

FIGURE 5. Latent web browser extension acquisition data types. Each data type can be configured for acquisition and will be sent from the web browser to the server over the Internet.

tab settings. All these sources can be configured, except browser and tab settings. The user has the flexibility to choose what to collect in any acquisition session (Fig. 5).

1) MOUSE INTERACTION

The mouse is the main computer peripheral used to interact with the web browser. For that reason, it can provide most of the relevant data from user browsing behaviour. The extension collects the mouse position on the browser window; the buttons clicked (left and right); and scroll behaviour associated with the mouse wheel. These data are collected within the context of five mouse events: mousemove, click, contextmenu, wheel and scroll. The mousemove event is triggered whenever the mouse moves, as the name indicates. Whenever any button is clicked the click event is triggered. The contextmenu event is triggered, when the user clicks the right mouse button (depending on the mouse configuration), or when pressing the context menu key on the keyboard. The wheel event is associated with the movement of the mouse wheel. The scroll event is not directly associated with the mouse. It is triggered when the page is scrolled, vertically or horizontally, by mouse or keyboard.

The mouse movement gives us an idea of the places the user moved through. It is known that there is some relation between the mouse pointer position and eye gaze [26]–[28]. Although this alignment may change under different circumstances, they are still generally related. This supports the choice of using mouse movement as an indicator of user focus. During the mouse movement the HTML hovered element XPath [29] is also collected (see Table 2). The XPath is only valid when the mouse hovers over a valid DOM element. On the case of Flash websites, it is not possible to gather this information since a Flash website is basically a kind of animation video, so there is no available DOM tree that represents most of the visible elements.

Data is collected via Javascript *addEventListener* method calls applied to the *document* object. Each call is attached to a specific event associated with the type of mouse interaction we want collect (see Table 2). Within the context of the *addEventListener* callback, the data is extracted from the *MouseEvent* object (passed to the callback) properties and sent to the web server.

2) KEYBOARD INTERACTION

The keyboard is the second main computer peripheral, and the main tool for text introduction. Evaluating the keyboard interaction is important to track user search profiles, form filling, and other actions associated with text insertion. Aside from the letters and numbers pressed, the Shift, Alt and Control keys, and Caps Lock are also tracked (see Table 2).

To ensure protection when entering data-sensitive username and password with the keyboard, data is never collected when the KeyDown event detected and password input event occurs. This prevents the acquisition of sensitive data that could comprise user security.

As stated in the *Mouse Interaction* sub-section, the data is also collected via a Javascript *addEventListener* method call applied to the *document* object. In this case, the event listened is *keydown*. Within the context of the *addEventListener* callback, the data is extracted from the *KeyboardEvent* object (passed to the callback) properties and sent to the web server.

3) GEOLOCATION

Geolocation is not browser navigation specific data but it is sometimes important to know the physical location of the user. This data is collected whenever a new browser tab is created. This type of data is subject to browser permissions approval on the extension installation (see Table 2).

The Javascript API used to collect the geolocation data is *window.navigator.geolocation* which returns a *Geolocation* object. The method *getCurrentPosition*, belonging to the

TABLE 2. Latent acquisition data types detailed information. The unit "px" correspond to pixels and the in italic format is identified each data type.

Interactions	Recorded events in detail	Needs user permission
Mouse	*Mouse move* - Mouse X,Y positions in px - *long* - Mouse X,Y positions with page left/right scroll in px - *long* - HTML element hovered XPath - *string* - Unix Timestamp - *int32* *Click* - Button code - *unsigned short* - Keyboard Shift, Alt and Control keys (pressed state) - *boolean* - Mouse X, Y positions in px - *long* - HTML element clicked XPath - *string* - Unix Timestamp - *int32* *Context menu* - Button code - *unsigned short* - Keyboard Shift, Alt and Control keys (pressed state) - *boolean* - HTML element clicked XPath - *string* - Unix Timestamp - *int32* *Wheel* - Delta Y (vertical scroll amount in px) - *long* - Unix Timestamp - *int32* *Scroll* - Scroll Top (distance in px from top of scrollbar to actual position) - *long* - Unix Timestamp - *int32*	No
Keyboard	*Keydown* - Char code - *unsigned long* - Keyboard Shift, Alt and Control keys (pressed state) - *boolean* - Caps Lock (on/off state) - *boolean* - Unix Timestamp - *int32*	No
Geolocation	*Page Load* - Altitude - *double* - Altitude accuracy - *double* - Latitude - *double* - Longitude - *double* - Latitude/Longitude accuracy - *double* - Heading - *double* - Speed - *double* - Unix Timestamp - *int32*	Yes
Screenshots	*Timer based (user setting)* - Base64 jpeg image data - *string* - Active tab ID - *int32* - Unix Timestamp - *int32*	Yes
Audio	*Timer based (sampling rate)* - 2 channel audio blob - *int32* - Sample rate - *float* - Buffer size of 2048 units of sample-frames (per channel) - *unsigned long* - Unix Timestamp - *int32*	Yes
Snapshots	*Timer based (user setting)* - Base64 jpeg image data (640x480px) - *string* - Unix Timestamp - *int32*	Yes
DOM	*Page Load & DOM change* - HTML document serialized string - *string* - Active tab ID - *int32* - Unix Timestamp - *int32*	No
Browser and Tab Settings	*Page Load* - Active tab ID - *int32* - Page URL - *string* - Browser window properties: width, height, avail width, avail height, orientation and color depth - *int, string* - User agent - *string* - Maximum number of touch points - *long* - Unix Timestamp - *int32*	No

previous object, is used to retrieve the geolocation data, which is sent to the server after the acquisition.

4) SCREENSHOTS

To better analyse the user behaviour and intentions while navigating, it is important to know what is being rendered on the screen (see Table 2). For that reason, active browser tab screenshots are collected every second, with the time interval being a setting that can be configured.

Because the main goal is to have a general idea of what the user is seeing, the quality of the JPEG image is low and pixelated. The criteria were to reduce the data size as much as possible but still allow the visual content to be perceived.

For screenshot capture, the Google Chrome tabs API (*chrome.tabs*) is used, more specifically, the *captureVisibleTab* method. Within its callback, the Base64 data is collected and sent to the server.

5) AUDIO

The audio collected from the microphone is not browser navigation specific data, but it can provide valuable insight into user behaviour and environment during web browsing (see Table 2). To be able to acquire audio, the user needs to manually give the browser media permissions.

The collection of audio data is based on the Javascript Web Audio API. A call to the method *getUserMedia* from *window.navigator.getMediaDevices* interface asks for user permission to collect audio media on the web browser, and retrieves a *Promise* that resolves to a *MediaStream* object. Within the context of the resolving callback a *window.AudioContext* object is created to which are attached to the media stream and a script processor node. Whenever there is audio data available on the buffer, the *onaudioprocess* event, belonging to the script processor node, is fired with the buffer data. A callback method is attached to that event where the data is gathered and sent to the server.

6) SNAPSHOTS

One of the data acquired is video camera snapshots every ten seconds (a time interval that can be configured). The snapshots are useful for research where the image allows to match facial expressions with keyboard or mouse behaviour. It can also be applied in usability tests or psychology experiments. See Table 2 for additional information.

To acquire the snapshots we use a combination of MediaDevices API and Canvas API. A call to the method *getUserMedia* from *window.navigator.getMediaDevices* interface asks for user permission to collect video media on the web browser, and retrieves a *Promise* that resolves to a *MediaStream* object. Within the context of the resolving callback, a canvas instance is created and a call to the *drawImage* method captures a snapshot of the video. Afterwards, the data is converted to Base64 and sent to the server.

7) DOCUMENT OBJECT MODEL (DOM)

According to [30], the ''Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page''. The DOM will be important to match XPath data with HTML document structure (see Table 2).

The DOM is acquired using the *serializeToString* method from the XMLSerializer interface, with the document object as the argument. This method returns an XML string representing a DOM tree. Taking into consideration the current Javascript ecosystem with all the frontend frameworks and asynchronous behaviours, we used the MutationObserver interface to watch for changes on the DOM tree and capture those changes. Whenever a change is detected the XML string is captured and sent to the server.

8) BROWSER AND TAB SETTINGS

During a page reload or tab creation, some settings are collected to allow a better definition of the conditions in which the browsing experience is running (see Table 2).

This settings are collected via standard Web APIs. The page URL is acquired via *window.location.href*; the browser settings use the *Window* and *Navigator* interfaces properties. The browser tab information is gathered through the *chrome.tabs* API.

C. DATA COLLECTION

After data acquisition, the Latent user needs to access the data stored on the web server. There are several options available, as long as the user knows how to program, but for the less computer savvy user, we present two simple options: using MongoDB's GUI, Compass; or using Latent's data extraction script.

Compass is MongoDB's standard GUI application and is very simple to use. It allows data browsing, visualization and exporting. It is possible to export the whole database or just some documents (MongoDB's table equivalent) to JSON or CSV formats. It can run on the server machine or any computer that can access the server remotely.

The data extraction script is a Python script that also exports the data to CSV, but it creates a separate file for each interaction. It runs on the server machine or any computer that can access the server remotely. The DOM data is not exported on a CSV file because the XML string breaks the column separation. Because of that, every DOM string is exported as a TXT file. The Audio data is exported in CSV and also as a WAV file.

IV. VALIDATION

The adherence to a new tool is dependent on its efficacy and efficiency. The nature of the Latent platform imposes some performance validation tests, on server requests, database

space needs and impact on computer performance, more specifically browser performance.

A. DATABASE SIZE

As previously stated, Latent generated data is stored in a NoSQL database. For proper system dimensioning it is important to have an idea of the hard drive space requirements for common data acquisition tasks. The data formats stored are audio blobs, base64 image data and general text. We shall analyse each format in more detail and show its disk space footprint.

1) AUDIO BLOB

The audio blobs consist of a list of numbers representing sound intensities. With a fixed sample rate of 44100 Hz, each sample takes about 2.268×10^{-5} seconds to acquire. A buffer of 2048 samples is being used, so it takes about 4.645×10^{-2} seconds to sample the whole buffer (blob) and send to the server. From this, we reach a value of about 21 blobs per second. Since each blob is about 0.02 *MB*, every second it is generated around 0.42 *MB* of audio data. For a clearer idea, using longer acquisition times we have:

- 1 *minute* : 25.2 *MB*
- 30 *minutes* : 756 *MB*
- 1 *hour* : 1512 *MB*

2) BASE64 IMAGE DATA

Regarding image data, Latent collects screenshots and snapshots, both as Base64 data. The default sampling rates for screenshots and snapshots are 1 Hz and 0.1 Hz, respectively.

The screenshots data size depends on the browser window size (which depends on the screen size for the upper limit), pixel density and colour palette. A sample of different screenshots taken from 50 random sites, using the browser in full-screen size, on a 23 inches screen with 1920×1080 pixels of resolution, was collected. We used the Random Website Machine[1] to browse through the random websites. The computed data size was, on average, 0.04 *MB*. The data size for various acquisition times are:

- 1 *minute* : 2.4 *MB*
- 30 *minutes* : 72 *MB*
- 1 *hour* : 144 *MB*

The snapshots data size also depends on the same characteristics as the screenshots. However, in this case, the image has a fixed size of 640×480 pixels. The image content will depend on who or what is behind the camera. For data size estimation, we collected around 70 samples with different configurations: single person, multiple persons, no persons, and different backgrounds. The computed data size was, on average, 0.05 *MB*. The data size for various acquisition times are:

- 1 *minute* : 3 *MB*
- 30 *minutes* : 90 *MB*
- 1 *hour* : 150 *MB*

[1] from whatsmyip.org [31]

3) STRUCTURED TEXT DATA

Within this category falls all other data stored as text. There are two main types: HTML DOM data and other key-value pair data.

The key-value pair data like the mouse, keyboard, geolocation, and settings require an average of 200 bytes per acquisition. The mouse is the one which produces more data since it creates a data sample for each movement.

The HTML DOM string is a lot more data intensive and depends on the page structure. This type of data is acquired every time a page loads or whenever the DOM is updated. To compare websites with distinct structures we collected data from six pages, apparently different in terms of elements and size. Google Search page and Youtube are two examples included as opposite cases.

The estimated data size for the different types of pages, that were merged into four groups similar in size, is:

- Light size[2]: ~ 0.03 *MB*
- Medium size[3]: ~ 0.25 *MB*
- Heavy size[4]: ~ 0.75 *MB*
- Very Heavy size[5]: ~ 1.1 *MB*

B. SERVER REQUESTS

In this section, we estimate the maximum number of requests (i.e. messages sent via the WebSockets connection) per second, that the extension sends to the server.

For every page load or reload, there are at least 5 requests sent to the server. Firstly, the web socketconnection (a.k.a. handshake) is established. Second, once this initial data is sent, four requests (messages) are sent with data: tab URL; browser settings; geolocation; and HTML DOM string. Finally, after all this initial data is sent, the extension starts listening for mouse, keyboard, and DOM update events and sending audio, screenshots and snapshots according to their acquisition rates. As previously stated, there are 21 audio blob requests per second, one screenshot request per second and one snapshot request every 10 seconds. Mouse and keyboard data do not have a fixed acquisition rate, hence not being easily quantifiable.

To quantify more precisely the maximum number of requests per second, we acquired data for 1 minute during some random navigation experience. The estimated value is the total number of requests during that time divided by the time of the experience. The results were approximately 42 requests per second.

Besides this, we can also evaluate possible limitations associated with the TCP/IP protocol, which allows $2^{16} = 65536$ connections per client IP. Since the web browser extension only establishes a maximum of 4 web socket connections with the server per tab at any given time, the web browser would need to have ~ 16384 opened tabs to reach the protocol connection limit. This large number allows us to comfortably

[2] https://www.tutorialspoint.com/
[3] https://www.google.com/
[4] https://www.youtube.com/
[5] https://www.google.com/search?q=content+heavy+sites&tbm=isch

assume that the browser extension will not exceed any TCP/IP protocol limitation.

To minimize possible lagging associated with data production/sending rates, several WebSockets are created to established dedicated communication channels for data-heavy interactions. There are separate WebSockets for audio, snapshots and screenshots. Mouse, keyboard, geolocation, DOM and settings, all share the same WebSocket.

C. BROWSER PERFORMANCE

To test how the extension affects the CPU usage percentage and available memory, tests scripts were developed using Python language. They consist of two threads running in parallel: one sample the computer system usage data every second, and the other runs the tests. The system utilization data were collected with the Python module *psutil*. It collected the CPU percentage, total virtual memory, and available virtual memory. The extension usage tests were created using Selenium Python and Chrome WebDriver to control Google Chrome.

Every test has the same structure. It starts with a five-minute acquisition of system resources data (*Pre-test* interval), followed by a five-minute test (*Test* interval), and ends with another five-minute of system resources data (*Post-test* interval). These two five-minute intervals exist to acquire the system baseline before and after the test. They also allow us to examine the browser activation/deactivation delays. During the *Test* interval, when the extension is active, the following interactions are being captured: Geolocation; Screenshots; Audio; Snapshots; DOM; and Browser and Tab settings. The tests consist of opening one or several web pages at the same time and capture the data associated with the interactions during the whole test period.

Each test is ran 30 times and we compute the mean signal for each characteristic and work with that signal for the analysis. For all tests the web server was running on a desktop PC with the following characteristics:

- Operating System: Ubuntu 18.04.1 LTS (64-bit)
- Processor: Intel® Core™ i5-6500 CPU @ $3.20GHz \times 4$
- Memory: 7.7 GB
- Graphics: Intel® HD Graphics 530 (Skylake GT2)

The computer running the browser had the following characteristics:

- Operating System: Ubuntu 16.04 LTS (64-bit)
- Processor: Intel® Core™ i5-6500 CPU @ $3.20GHz \times 4$
- Memory: 7.7 GB
- Graphics: GeForce GTX 960/PCIe/SSE2
- Web Browser: version 70.0.3538.67 (Official Build) (64-bit)
- Chrome Driver: version 2.43
- Python: version 3.6.5
- Selenium Python: version 3.14.1

TABLE 3. Web browser test statistical features. The features are extracted from the average signal of the 30 test runs.

Interval	Feature	CPU Usage (%)	Available Memory (%)
Pre-test	Mean± Std	0.2± 0.0	85.5± 0.0
	Median	0.2	85.5
Test	Mean± Std	0.3± 1.6	83.7± 0.0
	Median	0.2	83.7
Post-test	Mean± Std	0.2± 0.2	85.4± 0.0
	Median	0.2	85.4

1) WEB BROWSING WITH EMPTY TAB

The first test consists of just starting the web browser with an empty tab and leaving it on for five minutes. The goal was to measure the impact of the browser itself on the system. The data gathered is presented in Fig. 6, 7 and Table 3.

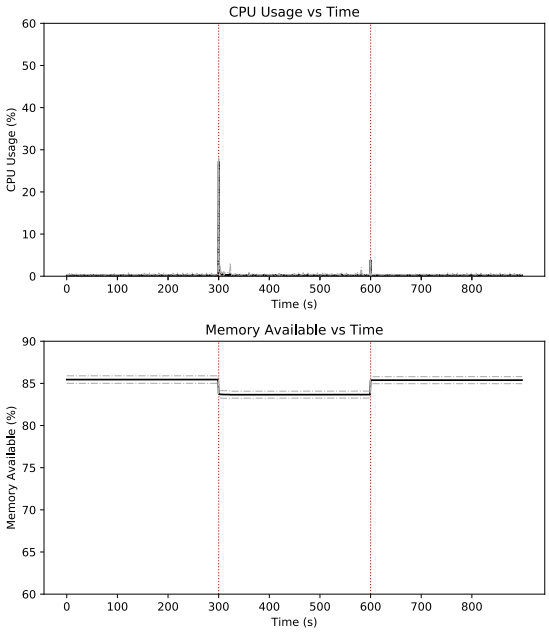

FIGURE 6. Web browser test plots for CPU usage (top) and available memory (bottom). The vertical lines define the limits of the intervals. The first and last correspond to the system resources baseline (*Pre-Test*, *Post-Test*), and the middle one corresponds to the static test (*Test*). The grey dashed-dotted lines represent the average signal of the 30 test runs, in black, plus or minus the standard deviation.

From Fig. 6 and 7 it is clear that the browser affects the system resources, but not much. The CPU usage has a spike on the moment the browser is opened but afterwards remains stable (less than 1%) with small fluctuations. The memory available decreases by 2% on average when the browser is opened, as shown in Table 3. These results provide a baseline behaviour of the system during the tests.

2) WEB BROWSING WITH ACTIVE TAB

In this case, the test starts the browser with one active tab having a web page loaded. It runs with and without the extension to see the effect of the extension on the performance.

TABLE 4. Web browser plus website test CPU usage statistics. The *On* and *Off* columns represent, respectively, tests using the Latent extension or not. The features are extracted from the average signal of the 27 test runs.

CPU Usage (%)											
Interval	Feature	Site 1		Site 2		Site 3		Site 4		Site 5	
		Off	On	Off	On	Off	On	Off	On	Off	On
Pre-test	Mean± Std	0.2± 0.1	0.1± 0.0	0.2± 0.0	0.1± 0.1	0.2± 0.0	0.1± 0.0	0.2± 0.0	0.1± 0.0	0.3± 0.2	0.1± 0.0
	Median	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.1
Test	Mean± Std	1.9± 2.7	16.6± 5.6	1.0± 3.1	16.2± 5.6	1.2± 2.7	16.0± 5.5	0.5± 2.4	15.0± 5.1	2.4± 3.1	16.5± 5.6
	Median	1.0	16.8	0.7	15.8	0.7	15.7	0.2	14.5	1.3	16.7
Post-test	Mean± Std	0.2± 0.3	0.11± 0.2	0.3± 0.3	0.2± 0.4	0.2± 0.1	0.1± 0.1	0.2± 0.1	0.1± 0.2	0.3± 0.3	0.1± 0.0
	Median	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.1

TABLE 5. Web browser plus website test memory available statistical features. The *On* and *Off* columns represent, respectively, tests using the Latent extension or not. The features are extracted from the average signal of the 27 test runs.

Memory Available (%)											
Interval	Feature	Site 1		Site 2		Site 3		Site 4		Site 5	
		Off	On	Off	On	Off	On	Off	On	Off	On
Pre-test	Mean± Std	80.7±0.0	73.2±0.0	80.6±0.0	73.1±0.0	80.6±0.0	73.1±0.0	80.5±0.0	73.0±0.0	80.5±0.0	73.0±0.0
	Median	80.7	73.2	80.6	73.1	80.6	73.1	80.5	73.0	80.5	73.0
Test	Mean± Std	78.1±0.1	68.6±1.5	77.5±0.1	68.0±1.6	77.9±0.1	68.5±1.4	78.3±0.0	68.9±1.3	78.2±0.0	68.8±1.3
	Median	78.1	68.1	77.5	67.2	77.9	67.8	78.3	68.6	78.1	68.4
Post-test	Mean± Std	80.6±0.2	73.1±0.0	80.6±0.2	73.0±0.0	80.5±0.2	73.0±0.0	80.5±0.2	73.0±0.0	80.4±0.1	72.9±0.0
	Median	80.6	73.1	80.6	73.1	80.5	73.0	80.5	73.0	80.4	72.9

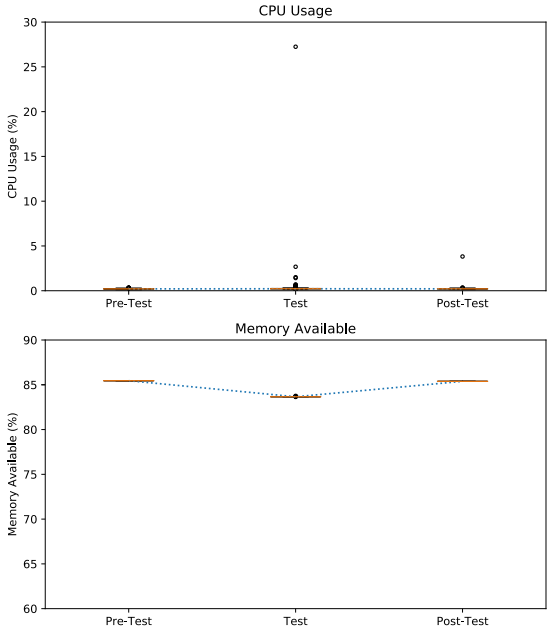

FIGURE 7. Web browser test box plots for CPU usage (top) and available memory (bottom). The *Pre-Test* and *Post-Test* data corresponds to the system resources baseline intervals data. The *Test* data refers to the test itself. The data is obtained in relation to the average signal of the 30 test runs.

Five different websites[6] are also used to measure how the extension's behaviour may be affected by web page structure and content. All the websites are dynamic and have large content. Two websites belong to the e-commerce category, another two are company websites, and the last one is a multimedia repository. For this test we only consider 27 runs as opposed to the established 30, because 3 tests were invalid. The data is presented in Fig. 10, 11, 12, and 13, as well as Tables 4 and 5. The figures only show data from two websites, the one with less variability (site 4) and the one with more variability (site 5), during the *Test* interval. The tables have the complete data for all websites.

Results show a stable behaviour between websites. Even considering specific variability, the CPU usage and memory available data follow the same patterns for every website on the three intervals, hence, we can consider the variability only between having the extension active and acquiring or not. Comparing Fig. 6 and 10, we can conclude that having a website loaded without the extension only consumes more memory and eventually has more CPU usage variability associated with specific website activities.

Comparing Fig. 10 and 11, it is clear that having the extension on produces an effect on the system performance. The CPU usage increases from an average of less than 3% to an average of around 16%. This result is expected because when the extension is acquiring data, many different activities start to consume CPU resources. However, an increase of roughly 15% is not significant. When examining the maximum values, we have around 30%. Even for those cases where the CPU usage still remains below 50%. Regarding the available memory, we see it decreasing during the test until a minimum value and then returning to the baseline value after the test is finished. The behaviour is no longer constant, but the difference between the baseline and the minimum value is approximately 6%. When the extension is off, this difference between baseline and the minimum value is around 3%. The extension consumes only about extra 3% memory. In this test scenario, the data shows that the extension has a relatively low impact on the available memory and a bigger impact on CPU usage, although this last one is not significant on average.

[6] Site 1: https://www.ebay.com
Site 2: https://www.youtube.com
Site 3: https://www.amazon.com
Site 4: https://www.apple.com
Site 5: https://www.microsoft.com

TABLE 6. Statistics for web browser usage with multiple sites and multiple tabs. The values are computed from the average signal of the 30 test runs.

Interval	Feature	CPU Usage (%)		Available Memory (%)	
		Off	On	Off	On
Pre-test	Mean± Std	0.2±0.0	0.2±0.3	87.4±0.0	85.8±0.0
	Median	0.2	0.1	87.4	85.8
Test	Mean± Std	5.5±5.3	16.5±5.6	79.9±1.2	81.4±1.2
	Median	3.8	16.7	79.5	81.1
Post-test	Mean± Std	0.3±0.6	0.2±0.6	87.1±1.3	85.8±0.4
	Median	0.2	0.1	87.3	85.8

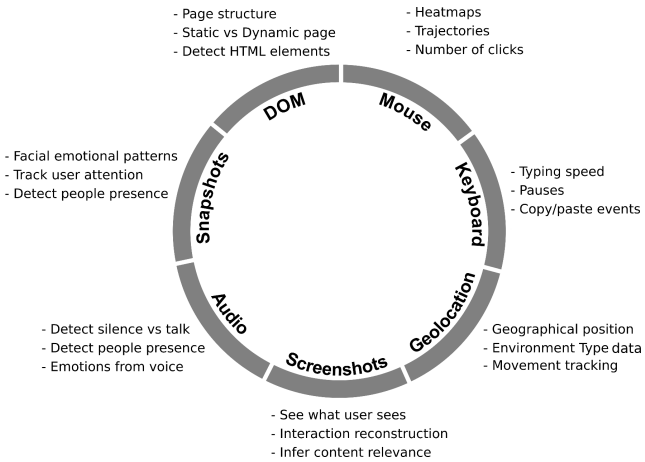

FIGURE 8. Examples of information that can be extracted from the acquired data.

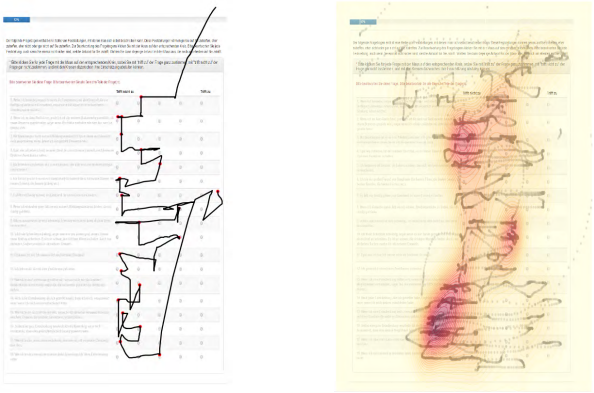

FIGURE 9. Examples of mouse visualization techniques. On the left the trajectory and on the right a heatmap.

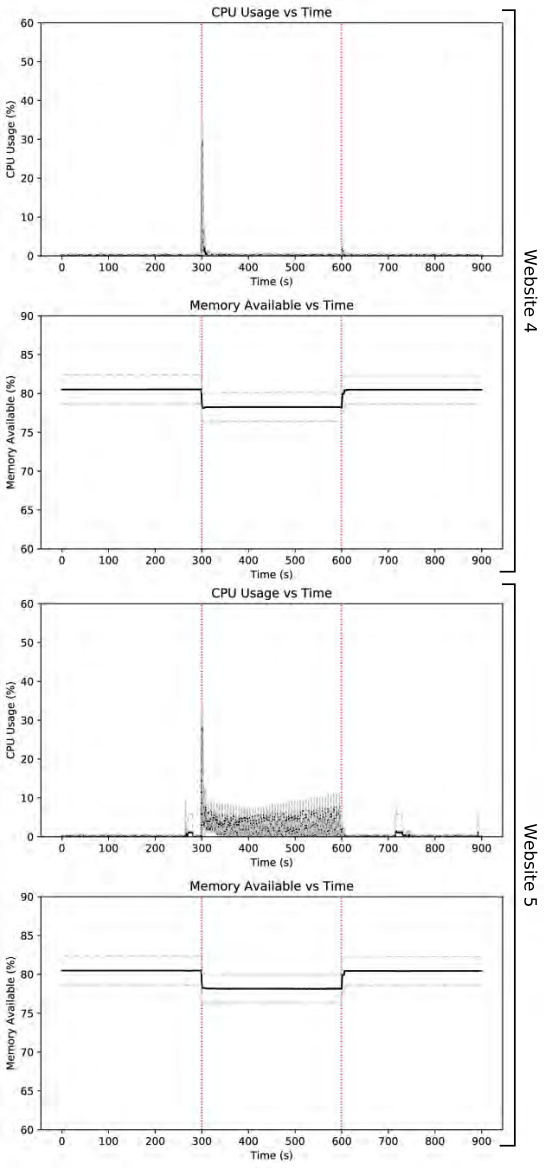

FIGURE 10. Web browser plus website test plots, without the use of the Latent browser extension, for CPU usage (first and third) and available memory (second and fourth). The vertical lines define the limits of the intervals. The first and last correspond to the system resources baseline (*Pre-Test*, *Post-Test*), and the middle one corresponds to the static test (*Test*). The first two graphs represent the data for the website with less variability. The last two graphs correspond to the website with more variability. The data corresponds to the average signal of the 27 test runs.

3) WEB BROWSING WITH MULTIPLE SITES AND MULTIPLE TABS

As new tabs are added to the web browser more system resources area used. This test measures how the extension may overload the system when using multiple tabs. Each tab is loaded with a different website using the Random Website Machine tool [31], to compensate the effects associated with web page structure and content. The test data is shown on Fig. 14 and 15, and Table 6.

Observing Fig. 14a (multiple tabs) and comparing it with Fig. 10a (one tab), it becomes clear that having more tabs by itself increases the CPU usage. The memory available also decreases when using multiple tabs (Fig. 14b) versus a single tab (Fig. 10b). Comparing between having the extension active versus not active, there is an expected difference in the system resources consumption (Fig. 14). The two plots on the left (Fig. 14a,b) are related to the test without the extension. The ones on the right (Fig. 14c,d) refer to the test with the extension. Regarding the CPU usage, with the extension active, there was a clear increase in relation to the test without the extension. On average, the maximum usage was around 40%. The grey lines indicate the standard

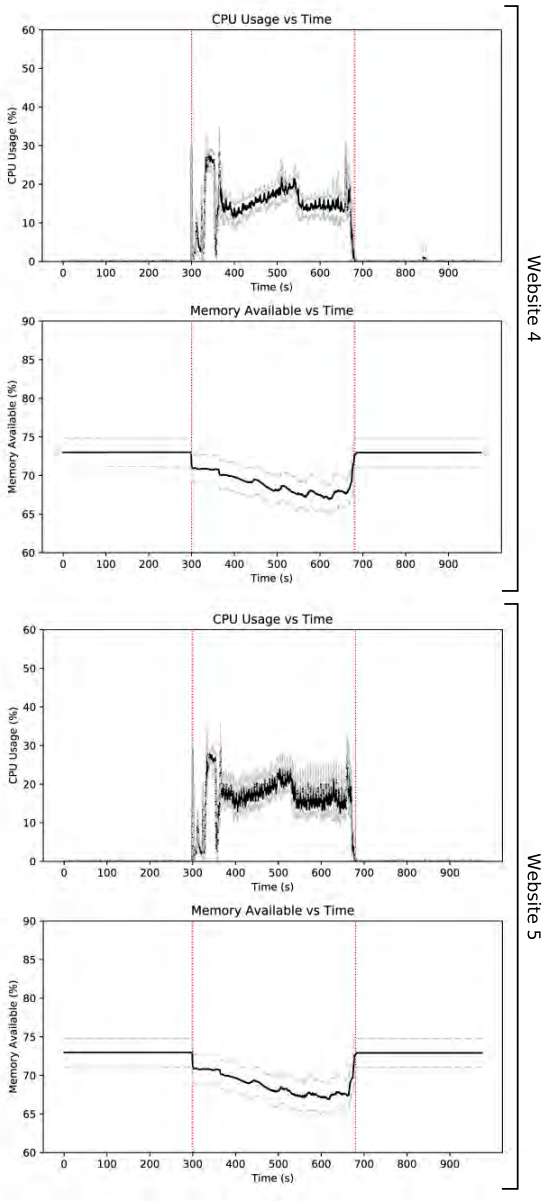

FIGURE 11. Web browser plus website test plots, using Latent browser extension, for CPU usage (first and third) and available memory (second and fourth). The vertical lines define the limits of the intervals. The first and last correspond to the system resources baseline (*Pre-Test*, *Post-Test*), and the middle one corresponds to the static test (*Test*). The first two graphs represent the data for the website with less variability. The last two graphs correspond to the website with more variability. The data corresponds to the average signal of the 27 test runs.

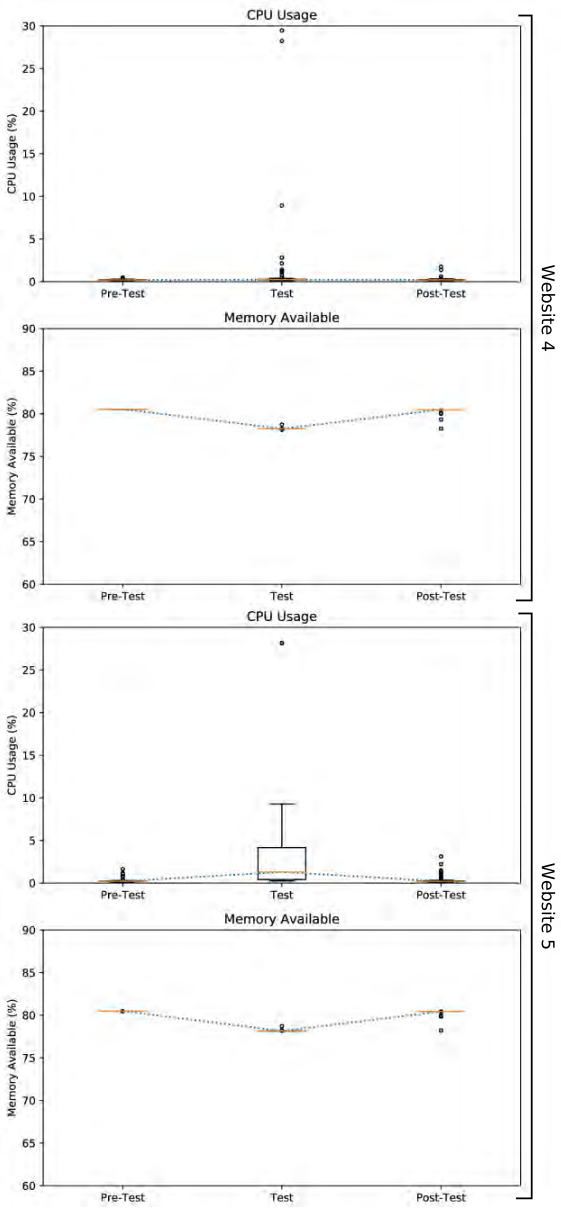

FIGURE 12. Web browser plus website test box plots, without the use of Latent browser extension, for CPU usage (first and third) and available memory (second and fourth). The vertical lines define the limits of the intervals. The first and last correspond to the system resources baseline (*Pre-Test*, *Post-Test*), and the middle one corresponds to the static test (*Test*). The first two graphs represent the data for the website with less variability. The last two graphs correspond to the website with more variability. The data corresponds to the average signal of the 27 test runs.

deviation for the 30 tests. We can see from those lines that, with or without the extension there was at least one test that had a peak event around 60%. However, having multiple tabs opened with the extension uses 16.5 ± 5.6 (%) of CPU, versus 5.5 ± 5.3 (%). It is an increase of around 11.2%. Looking at the available memory (Fig. 14b vs. Fig. 14d), we see a change on the behaviour of memory consumption. Without the extension, the memory available decreases sharply when the browser is open with multiple tabs and remains fairly constant during the whole time until the browser is closed. When the extension is used, there is a first step associated with the activation of the extension acquisition. Afterwards, the memory continues to decrease slowly until a minimal value. The difference between the maximum value during *Pre-test* and the minimum value during *Test* intervals is about 6% and 8%, with the extension and without, respectively. As before, we can see that the presence of the extension does not have a significant impact on memory consumption.

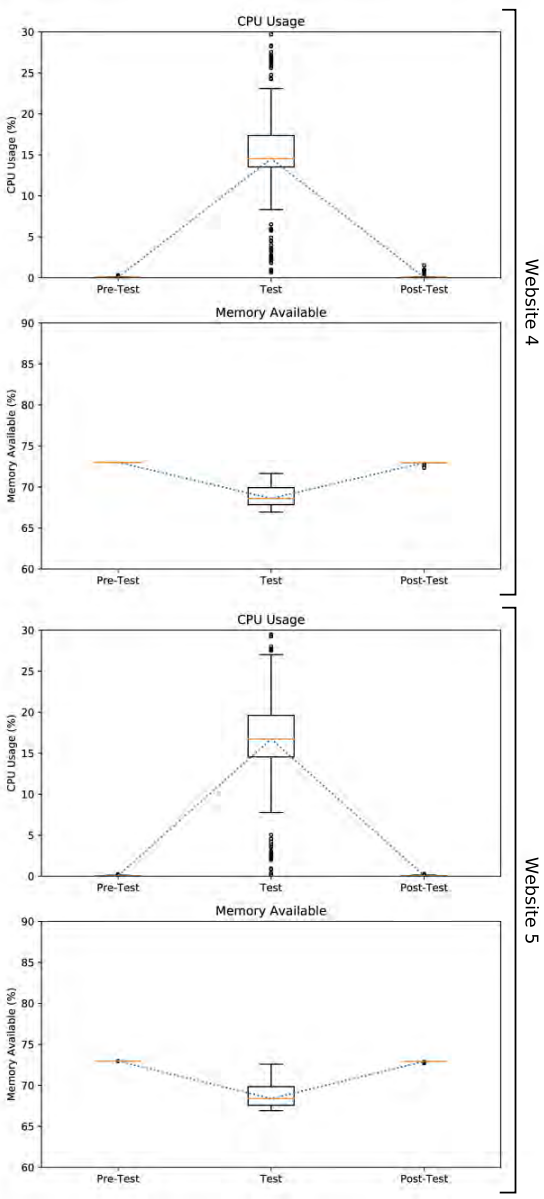

FIGURE 13. Web browser plus website test box plots, using Latent browser extension, for CPU usage (first and third) and available memory (second and fourth). The vertical lines define the limits of the intervals. The first and last correspond to the system resources baseline (*Pre-Test, Post-Test*), and the middle one corresponds to the static test (*Test*). The first two graphs represent the data for the website with less variability. The last two graphs correspond to the website with more variability. The data corresponds to the average signal of the 27 test runs.

V. USE CASES

The Latent platform flexibility resides on the sheer volume of different data that it can acquire. This diversity opens up the possibility for many different data visualizations and processing techniques. This section presents some of the possible visualization techniques and information extracted from the data acquired, as well as the different scenarios where we see advantages on the use of Latent.

A. DATA VISUALIZATION TECHNIQUES & INFORMATION EXTRACTED

Latent-acquired data allows the generation of commonly used visualization strategies like Heatmaps, Scrollmaps or Spatial Clicks Distribution. For example, with the data acquired during the mouse move event, it is possible to create mouse movement trajectories, analyse movement patterns, and evaluate aspects, such as speeds and accelerations. With the screenshot data, we have the ability to reconstruct the browsing session, estimate the time focused on a certain area of the web page, etc. Fig. 8 shows some of the information that is possible to extract from the various data types and in Fig. 9 are represented two visualization examples.

B. APPLICATION AREAS

Based on the diverse range of information, we can extract from the data acquired by Latent, we propose the following application areas:

- **Web usability tests**: With the data acquired it is possible to get the typical information used in usability tests [32]–[34], allowing the user to test web pages or web applications.
- **Decision-making process analysis**: In some scenarios, it is more important to evaluate the thought process as opposed to the result. If an experience is created on the web [35], Latent is able to provide data to support the analysis of the whole process from beginning to end.
- **Psychology studies**: This field of research is probably one that is most likely to benefit from this extension. Analysis of behaviour and personality are good examples [36], [37]. This can include measures that have received less attention, such as the evaluation of mouse behaviour, which has some potential as an indicator of personality traits.
- **Human resources screening tests**: Personal assessment by recruitment agencies and employers, to apply, e.g, technical tests as part of the applicant assessment process. If these tests are applied on in online format (like [38], [39]), it is possible to use Latent to extract additional information about the applicant's problem solving approach, steps, or strategy.
- **Serious games**: These type of games are used in many different contexts, from education to health. In health, serious games are sometimes used for behavioural analysis in people with mental or learning disabilities [40]. Latent could provide a deeper insight into the game play dynamics.
- **Survey analysis**: In cases where it is important to understand how people fill surveys [12]. The collected data may be used to infer personality traits, or to analyse how the structure of the survey affects the way that the survey respondent fills out the survey. Latent is flexible enough to extract the necessary data for the analysis.

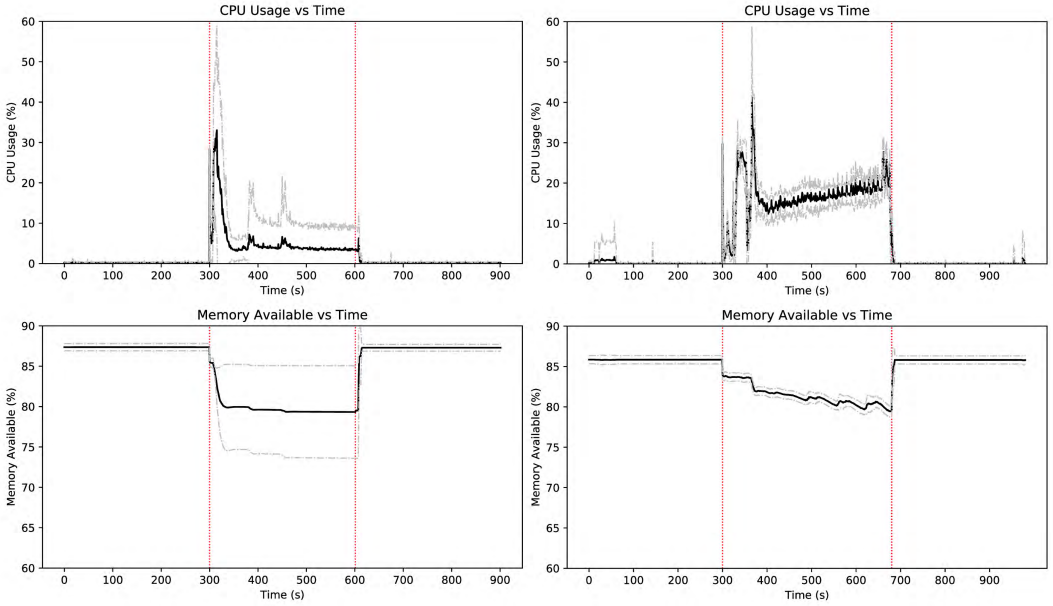

FIGURE 14. Web browser with multiple sites and multiple tabs test plots for CPU usage (top) and available memory (bottom). The left plots correspond to tests performed without using the extension, and right plots correspond to tests done with the extension.

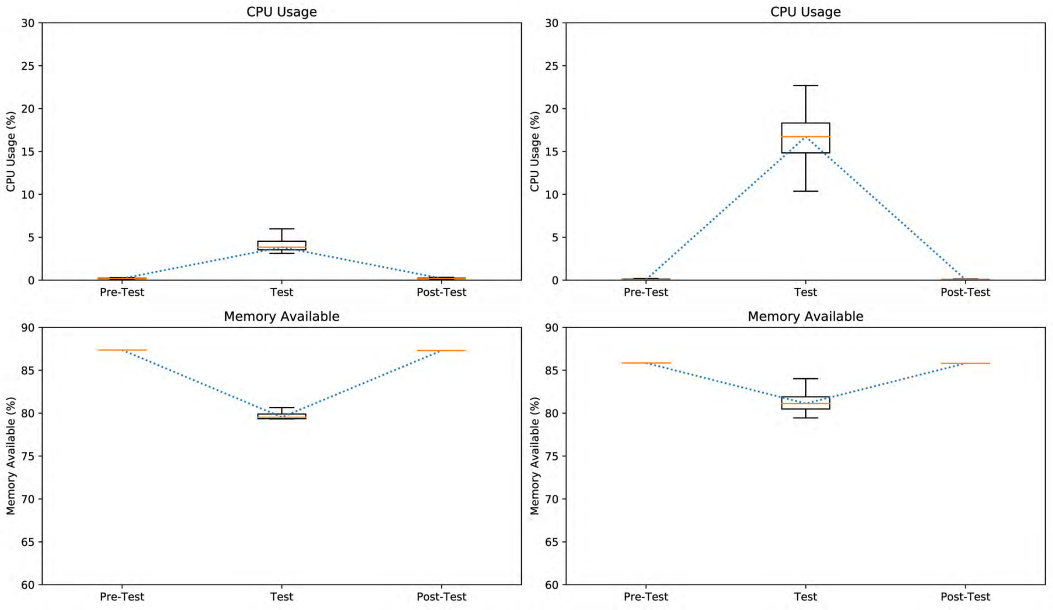

FIGURE 15. Web browser with multiple sites and multiple tabs test box plot for for CPU usage (top) and available memory (bottom). The left plots correspond to tests performed without using the extension, and right plots correspond to tests done with the extension.

VI. CONCLUSION

This paper presents the details and functionalities of a new tool to monitor web browsing behaviour and its performance. The main goal of this tool is to collect all possible information about the user and his/her interaction with the computer while interacting with the web. In this field, the existing services have restricted use and the raw data is not usually available for further investigation. Latent provides the necessary flexibility for customized data analysis, allowing a large number of possible use cases to be derived from the data collected.

In terms of performance, the impact of the platform on the system's architecture was assessed based on database size, server requests and browser performance. This performance analysis also serves the estimation of costs associated with the resources consumption.

As expected, the audio and images, that is, screenshots and snapshots, are data types requiring more space, resulting in approximately 1.7 GB during one hour of data acquisition. The data types that do not have a defined acquisition frequency, such as the keyboard, require storage space that depends on the specific user interaction. In relation to the server load, the requests are below 50 per second.

The browser performance was tested by assessing the CPU and memory consumption while using the browser extension. For a more general assessment, the trials were done on a single tab or multiple tabs configuration, using distinct websites for each test/tab. As expected, there was an increase in the CPU usage and memory consumption, although not to a significant level which could compromise the user interaction.

In a next step, this tool should be improved and tested. For example, some level of data compression should be applied to reduce memory consumption. A real scenario test with a detailed procedure should be considered to evaluate the platform performance and errors when analysing, for example, different ways, or user habits and preferences, of browsing the web. This should also be assessed for multiple simultaneous users of the platform.

ACKNOWLEDGMENT

Web browser, web server and database symbols made by *Smashicons* from www.flaticon.com. Internet cloud symbol made by *itim2101* from www.flaticon.com. Websockets logo was downloaded from worldvectorlogo.com. *(Catia Cepeda and Ricardo Tonet contributed equally to this work.)*

REFERENCES

[1] Internet World Stats. (2017). *Internet Growth Statistics 1995 to 2017—The Global Village Online*. [Online]. Available: https://www.internetworldstats.com/emarketing.htm

[2] Google. (2018). *Google Search Statistics—Internet Live Stats*. [Online]. Available: http://www.internetlivestats.com/google-search-statistics/

[3] E. Hehman, R. M. Stolier, and J. B. Freeman, "Advanced mouse-tracking analytic techniques for enhancing psychological science," *Group Processes Intergroup Relations*, vol. 18, no. 3, pp. 384–401, 2015. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1368430214538325

[4] B. De Los Santos and A. Hortaçsu, "Testing models of consumer search using data on Web browsing and purchasing behavior," *Amer. Econ. Rev.*, vol. 102, no. 6, pp. 2955–2980, Oct. 2012. [Online]. Available: www.netinst.org

[5] R. E. Bucklin and C. Sismeiro, "A model of Web site browsing behavior estimated on clickstream data," *J. Marketing Res.*, vol. 40, no. 3, pp. 249–267, 2003. [Online]. Available: http://journals.sagepub. com/doi/10.1509/jmkr.40.3.249.19241 and http://journals.ama.org/doi/abs/10.1509/jmkr.40.3.249.19241

[6] P. Shuxin, J. Fan, S. Yu, B. Wang, X. Jia, R. Hu, and Q. Zhu, "A method of behavior evaluation based on Web browsing information," in *Proc. Int. Conf. Smart Grid Electr. Autom. (ICSGEA)*, 2017, pp. 697–700. [Online]. Available: http://ieeexplore.ieee.org/document/8104482/

[7] Deepika, S. Juneja, and A. Dixit, "Improving search results based on users' browsing behavior using *Apriori* algorithm," in *Advances in Intelligent Systems and Computing*, vol. 731. Singapore: Springer, 2019, pp. 73–82. [Online]. Available: http://link.springer.com/10.1007/978-981-10-8848-3_7

[8] C. Doucet and R. M. Stelmack, "Movement time differentiates extraverts from introverts," *Personality Individual Differences*, vol. 23, no. 5, pp. 775–786, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191886997001049

[9] M. Kosinski and D. Stillwell, "Personality and Website choice," in *Proc. ACM Web Sci. Conf.*, 2012, pp. 251–254. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.296.5575&rep=rep1&type=pdf

[10] C. Li, J. Wan, and B. Wang, "Personality prediction of social network users," in *Proc. 16th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci. (DCABES)*, Sep. 2018, pp. 84–87. [Online]. Available: http://ieeexplore.ieee.org/document/8253041/

[11] I. A. Khan, W.-P. Brinkman, N. Fine, and R. M. Hierons, "Measuring personality from keyboard and mouse use," in *Proc. 15th Eur. Conf. Cognit. Ergonom. Ergonom. Cool Interact. (ECCE)*, 2008, p. 1.

[12] C. Cepeda, J. Rodrigues, M. C. Dias, D. Oliveira, D. Rindlisbacher, M. Cheetham, and H. Gamboa, "Mouse tracking measures and movement patterns with application for online surveys," in *Lecture Notes in Computer Science* (Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11015. Cham, Switzerland: Springer, 2018, pp. 28–42. [Online]. Available: http://link.springer.com/10.1007/978-3-319-99740-7_3

[13] *ViewletBuilder—Easily Create Flash Tutorials and Simulations Using Screen Capture Technology*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.qarbon.com/presentation-software/viewletbuilder/

[14] *Get GIF Camcorder—Microsoft Store*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.microsoft.com/en-us/p/gif-camcorder/9wzdncrdxxwr?activetab=pivot:overviewtab

[15] *Crazy Egg Website Optimization|Heatmaps &Amp; A/B Testing*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.crazyegg.com/

[16] *Inspectlet—Session Recording, Website Heatmaps, Javascript A/B Testing, Error Logging, Form Analytics*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.inspectlet.com/

[17] *Ferramenta Gratuita de Gravaç ão Para Websites e Aplicativos|Smartlook*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.smartlook.com/pt/

[18] *Hotjar—Heatmaps, Visitor Recordings, Conversion Funnels, Form Analytics, Feedback Polls and Surveys in One Platform*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.hotjar.com/home1?utm_expid=.Z6cvEiV7SFmTDSpI1b1J-A.1&utm_referrer=https%3A%2F%2Fwww.google.pt%2F

[19] *Session Replay, Heatmaps, Funnels, Forms &Amp; User Feedback—Mouseflow*. Accessed: Nov. 28, 2018. [Online]. Available: https://mouseflow.com/?gclid=CjwKCAiAlvnfBRA1EiwAVOEgfHnRX-MXU9pACopLxprF8J–l46nlvUEOUmGrE9HxDlXmXx6U9N0WhoCGggQAvD_BwE

[20] *Enterprise Experience Analytics|Conversions Optimization|Clicktale*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.clicktale.com/

[21] *HYFY Screen Video Recorder*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.hyfy.io/

[22] *ReMouse—Mouse Recorder, Keyboard Recorder, GhostMouse, Auto Clicker, AutoClick, Auto Mouse*. Accessed: Nov. 28, 2018. [Online]. Available: https://www.remouse.com/

[23] H. F. S. Gamboa, "Multi-modal behavioral biometrics based on HCI and electrophysiology," Ph.D. dissertation, Dept. Elect. Comput. Eng., Instituto Superior Técnico, Lisbon, Portugal, 2008. [Online]. Available: http://www.lx.it.pt/~afred/pub/thesisHugoGamboa.pdf

[24] *Tornado Web Server—Tornado 6.0.2 Documentation*. Accessed: Jun. 17, 2019. [Online]. Available: http://www.tornadoweb.org/en/stable/#

[25] A. Lombardi, *WebSocket: Lightweight Client-Server Communications*. Sebastopol, CA, USA: O'Reilly Media.

[26] J. Huang, R. White, and G. Buscher, "User see, user point: Gaze and cursor alignment in Web search," in *Proc. ACM Annu. Conf. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2012, p. 1341. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2207676.2208591

[27] U. Demšar and A. Çöltekin, "Quantifying gaze and mouse interactions on spatial visual interfaces with a new movement analytics methodology," *PLoS ONE*, vol. 12, no. 8, p. e0181818, 2017. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0181818

[28] D. J. Liebling and S. T. Dumais, "Gaze and mouse coordination in everyday work," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. Adjunct Publication UbiComp Adjunct*, New York, NY, USA, 2014, pp. 1141–1150. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2638728.2641692

[29] *XPath|MDN*. Accessed: Mar. 06, 2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/XPath

[30] World Wide Web Consortium. (2006). *W3C Document Object Model*. [Online]. Available: https://www.w3.org/DOM/ and http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:W3C+Document+Object+Model#2

[31] *Random Website Machine/WhatsMyIP.org*. [Online]. Available: http://www.whatsmyip.org/random-website-machine/
[32] E. Arroyo, T. Selker, and W. Wei, "Usability tool for analysis of Web designs using mouse tracks," in *Proc. Extended Abstr. Hum. Factors Comput. Syst. (CHI EA)*, New York, NY, USA, 2006, p. 484. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1125451.1125557
[33] N. Nakamichi, K. Shima, M. Sakai, and K.-I. Matsumoto, "Detecting low usability Web pages using quantitative data of users' behavior," in *Proc. 28th Int. Conf. Softw. Eng. (ICSE)*, New York, NY, USA, 2006, p. 569. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1134285.1134365
[34] M. M. Rahman and N. A. Abdullah, "A personalized group-based recommendation approach for Web search in E-learning," *IEEE Access*, vol. 6, pp. 34166–34178, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8395492/
[35] A. Yanie, A. Hasibuan, I. Ishak, M. Marsono, S. Lubis, N. Nurmalini, M. Mesran, S. D. Nasution, R. Rahim, H. Nurdiyanto, and A. S. Ahmar, "Web based application for decision support system with ELECTRE method," *J. Phys., Conf. Ser.*, vol. 1028, no. 1, p. 012054, 2018. [Online]. Available: http://stacks.iop.org/1742-6596/1028/i=1/a=012054?key=crossref.bc11a878192249389a76d1cfa8d74f33
[36] M. C. Chen, J. R. Anderson, and M. H. Sohn, "What can a mouse cursor tell us more?" in *Proc. Extended Abstr. Hum. Factors Comput. Syst. (CHI)*, New York, NY, USA, 2003, p. 281. [Online]. Available: http://portal.acm.org/citation.cfm?doid=634067.634234
[37] M. Graff, "Individual differences in hypertext browsing strategies," *Behav. Inf. Technol.*, vol. 24, no. 2, pp. 93–99, 2005. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01449290512331321848
[38] B. Pfieffelmann, S. H. Wagner, and T. Libkuman, "Recruiting on corporate Web sites: Perceptions of fit and attraction," *Int. J. Selection Assessment*, vol. 18, no. 1, pp. 40–47, 2010. [Online]. Available: http://doi.wiley.com/10.1111/j.1468-2389.2010.00487.x
[39] U. D. Reips and R. Lengler, "The*Web Experiment List*: A Web service for the recruitment of participants and archiving of Internet-based experiments," in *Behavior Research Methods*, vol. 37. New York, NY, USA: Springer-Verlag, 2005, no. 2, pp. 287–292. [Online]. Available: http://www.springerlink.com/index/10.3758/BF03192696
[40] L. Martini, F. Vannetti, L. Fabbri, F. Gerli, I. Mosca, S. Pazzi, F. Baglio, and L. Bocchi, "GOAL (games for olders active life): A Web-application for cognitive impairment tele-rehabilitation," in *IFMBE Proceedings*, vol. 68. Singapore: Springer, 2019, no. 3, pp. 177–182. [Online]. Available: http://link.springer.com/10.1007/978-981-10-9023-3_32