HCI Browser: A Tool for Administration and Data Collection for Studies of Web Search Behaviors

Robert Capra

School of Information and Library Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, USA
rcapra3@unc.edu

Abstract. We describe the HCI Browser, a Mozilla Firefox extension designed to support studies of Web information seeking. The HCI Browser presents configurable tasks to the user, collects browser event data as the user interacts with the browser and Web pages, provides mechanisms to record answers that are found, and administers pre- and post-task questionnaires. In this paper, aspects of using and configuring the HCI Browser are summarized and details are given about the events logged, the format of the log files, and how the system is implemented as a Firefox extension. The HCI Browser is open-source software and is available for download at: <http://ils.unc.edu/hcibrowser>

Keywords: Web information seeking, user interface event logging, data collection.

1 Introduction

Laboratory studies of Web information seeking often involve presenting tasks to participants and recording their actions and behaviors as they use a Web browser to search for information requested by the tasks [4]. Researchers have built and used a variety of tools to help support data collection for Web search and information seeking tasks [1][6][7][8][9][10][11][13][14] and have noted the challenges involved with capturing naturalistic user behaviors [10]. We reviewed a number of approaches [4] and found that they did not meet the needs we had for administering laboratory studies. Specifically, we needed a tool that would: 1) integrate with an existing Web browser to provide a familiar browsing experience to participants, 2) record a wide variety of user interactions with the Web pages and the browser interface, and 3) provide support for administrative aspects of conducting a laboratory study (e.g. administration of questionnaires, closing windows from completed tasks) [4]. To support these needs, we developed the HCI Browser, drawing upon our previous experience building an instrumented web browser [6] and also utilizing concepts from the open-source Lemur Query Log Toolbar project [11]. We have described aspects of the HCI Browser elsewhere [3][4][5]. In this paper, we summarize the functions and operation of the HCI Browser (similar to aspects discussed in [3][4][5]), and extend our prior work by describing in detail the events and data that are logged. We

also provide detailed log file examples with explanations and include information about how the HCI Browser is implemented.

The HCI Browser is released as open-source, “alpha” level software. As such, it may contain bugs and we remind potential users to conduct their own testing. Since it is released as open-source software, others can make modifications and improvements to the code.

2 HCI Browser Walk-Through

The HCI Browser is installed as an extension to the Mozilla Firefox Web browser. Because the HCI Browser modifies the Firefox interface and displays pop-up windows and dialog boxes every time Firefox is run, it is recommended to install the HCI Browser into a separate Firefox user profile that is used only for data collection studies. The HCI Browser is designed to: present experimental tasks to users, collect and log browser event data, automatically manage the opening and closing of windows as needed, and to administer optional pre- and post-task questionnaires [4]. In this section, we present the basic functions of the HCI Browser. In the following section, we describe configuration options.

2.1 Start-Up

After installing the HCI Browser extension, every time Firefox is started, a dialog box similar to Fig. 1(a) will be displayed. This dialog box allows the experimenter to enter information that will be used to label data that is logged for the session, including a session number, participant id, and starting task number. Clicking “OK” will advance to a configurable introduction screen, shown in Fig. 1(b), which can be used to display information and instructions to study participants. Clicking “OK” on the introduction screen will advance to an optional pre-task questionnaire screen, or will go directly to the first task if no questionnaire has been configured.

Fig. 1. (a) Start-up dialog box, and (b) Introduction screen

2.2 Pre-task and Post-task Questionnaires (Optional)

The experimenter can design optional pre- and post-task questionnaires to be displayed before and after the participant completes each task. Examples of these are shown in Fig.2 (a) and (b). Three types of questions are supported: multiple choice (responses listed vertically), Likert-type/semantic differential (responses listed horizontally), and open answer (text box). The responses that participants provide are logged and labeled using the information entered by the experimenter on the start-up screen. The task number and prompt text are displayed at the top of the questionnaire screens so that the participant can keep the task context in mind while answering. Note that if configured, the questionnaires are presented before and after **each** task. Different questions can be configured for the pre-task and post-task questionnaires.

Fig. 2. (a) Pre-task questionnaire, and (b) Post-task questionnaire

2.3 Task Presentation and Answer Collection

After the participant has answered any pre-task questions, the HCI Browser will automatically close the questions window and display a normal Firefox browser window with a toolbar area added above the browser tabs. This section of the window is shown in Fig. 3. In the toolbar area, the task number and task prompt are displayed, along with information about how many answers can be submitted for this task (configurable) and how many answers have been submitted by the user so far. Users can search, browse, and navigate the Web as normal using the browser to look for the information requested by the task. As users work with the browser, their actions are logged as described later in this paper. When they find particular information that they would like to identify as an “answer” for the information requested by the task, they can click the “Found an answer on this page” button in the toolbar. Upon doing so, the right side of the toolbar display changes as shown in Fig. 4 to record the URL of the current web page and to provide a place (labeled “answer text”) for the participant to type (or copy and paste) text from the page that is relevant to the task.

When the participant is satisfied with the answer, they can click the “Submit this answer” to continue to look for more answers for the task or they can finish the task by clicking the “Done with this task” button. Tasks can be configured to allow users to submit a maximum number of answers as a method to help set bounds on the amount of time and information requested.

Fig. 3. Toolbar area displaying the task and answer controls

Fig. 4. Toolbar area displaying the answer recording controls

After the participant finishes finding answers for the task (i.e. clicks the “Done with answers for this task” button or submits the maximum number of answers), the HCI Browser will automatically close all open browser windows and display the post-task questionnaire for the task, if it has been configured. If not, then the system will move on to the next task. After completing the last task, the system displays a message letting the participant know that they have completed all the tasks.

3 Configuring the HCI Browser

Configuration of the HCI Browser is accomplished by editing configuration files located in a folder called “*hcibrowser*” on the operating system Desktop. There are four main configuration files: *intro.txt*, *pretask.txt*, *posttask.txt*, and *tasks.txt*. These files are all simple text files that can be edited using any text editor.

The *intro.txt* file contains the text to be displayed on the Introduction screen shown in Fig. 1(b). The *pretask.txt* and *posttask.txt* files are used to configure the question types and question text presented in the pre- and post-task questionnaires respectively. If these either of these files is not present in the “*hcibrowser*” configuration folder on the Desktop, then the respective questionnaire will not be presented to the participants. The format for the *pretask.txt* and *posttask.txt* files is identical. Fig. 5 shows the *pretask.txt* file that corresponds to the questionnaire in Fig. 2(a).

```
01 MultipleChoice
02 5
03 How often do you look for information about this topic?
04 Daily
05 Weekly
06 Monthly
07 Yearly
08 Never
09 ---
10 LikertType
11 5
12 How difficult do you think the task will be?
13 Very easy
14
15
16
17 Very difficult
18 ---
19 OpenAnswer
20 What strategy do you plan to use to look for the information?
21 ---
```

Fig. 5. Example pre-task questionnaire configuration file. Line numbers are shown here for description and are not included in the actual configuration file.

Three types of questions are supported:

- Multiple choice (Fig. 5, lines 01-09) – response choices are displayed vertically. The keyword “MultipleChoice” is used in the configuration file (line 01), followed by a line indicating how many choices there will be (line 02), a line for the question text (line 03), followed by the text of the choices, one choice per line (lines 04-08). A required line of three minus signs (line 09) indicates the end of the question.
- Likert-type/semantic differential (Fig. 5, lines 10-18) – response choices are displayed horizontally. The keyword “LikertType” is used in the configuration file (line 10), followed by a line indicating how many choices there will be (line 11), a line for the question text (line 12), followed by the text of the choices, one choice per line (lines 13-17). In this example, lines 14-16 are blank because only endpoint anchor text was desired for this scale. Line 18 is the required question terminator line.
- Open answer (Fig. 5, lines 19-21) – the question is presented along with a text box for the participant to type an open response. The keyword “OpenAnswer” is used in the configuration file (line 19), followed by a line for the question text (line 20). Line 21 is the required question terminator line of three minus signs.

The final configuration file is named *tasks.txt* and contains information about the tasks to be presented. An example is shown in Fig. 6. Each task to be presented requires two lines in the configuration file: an initial line that contains the text of the task prompt (e.g. lines 01, 03, and 05 in Fig. 6), and a second line that contains the maximum number of answers that will be accepted for the task (e.g. lines 02, 04, and 06). The tasks will be automatically numbered and presented in the order in which they appear in the *tasks.txt* configuration file.

```

01 How tall is the U.S. capital building in Washington, DC?
02 1
03 What is being done to help reduce childhood obesity in the U.S.?
04 3
05 What are some reported causes of global warming?
06 3

```

Fig. 6. Example *tasks.txt* configuration file. Line numbers are shown here only for description and are not included in the actual configuration file.

4 Events and Data Logging

Events and data collected while running the HCI Browser are logged into an automatically created sub-folder called “*data*” in the “*hcibrowser*” folder on the Desktop. Within the “*data*” folder, additional sub-folders are created for each session and participant id entered at the start-up screen. Individual text-based log files are placed in these folders for each task and are named based on the session, participant id, task number, and time stamp (e.g. S42_P14_T2_1297118649034.txt). The log files contain data from the pre- and post-task questionnaires (if administered) and from the participant’s interactions with the browser during the task.

Events logging is based on the HCI Browser observing Firefox browser events, filtering and interpreting these events, and writing them to a log file. Many of the events used by the HCI Browser are similar to underlying Firefox events [12] and also to event labels used by the Lemur Query Log Toolbar project [11]. Table 1 lists a subset of common events that are logged. Additional events are logged for actions that involve the browser history and bookmark menus.

5 Understanding the Log Files

The log files are organized as a text file with one event per line. Information on each line is separated by tab characters (i.e. tab delimited), making it easy to read the log file into a spreadsheet or other analysis program. Fig.7 shows an example line of a log file. The first eight fields on each line are fixed and contain information as follows. The first field is a date+time stamp in numeric format (1297118695677), followed by a human-readable date (7-2-2011 = Feb 7, 2011), and time (17:44:55 = 5:44pm and 55 seconds).

The next three fields are the session number (S42), participant id (P14), and the task number (T2). The next field indicates if the current line was logged during the pre- or post-task questionnaires (“pretask” and “posttask”), or while the participant was doing the task itself (“intask”). The final fixed field indicates the event and uses codes as outlined in Table 1.

Table 1. Events logged by the HCI Browser

Event	Event code	Description
Page load	LoadCap	A page is loaded into a window or tab. This event does not mean that the page is being viewed (i.e. in focus). Based on the Firefox “load” event.
Focus	Focus	The focus has changed, meaning that the user is now viewing a different tab or window than they were before the focus change. Based on the Firefox “focus” event.
Scrolling	Scroll	Scrolling on the page. This event is logged based on a timer such that subsequent scrolling is not logged for 500 milliseconds.
Left click, right click	LClick, RClick	The user has clicked the mouse on a page element. Left-clicks typically are link clicks. Right-clicks often are to open a link in a new tab or window via the right-click context menu.
Add, select, remove tab	AddTab, SelTab, RmTab	Actions dealing with browser tabs. Also logs a set of informational messages listing currently open browser windows and tabs. SelTab typically also results in a focus change event.
Information	Info	Informational messages (e.g. list of open tabs)
Typing URLs	TypeURL	The user has typed into the browser’s address bar (e.g. typed in a URL to navigate to it)
Browser button clicks	[Home Back Forward Reload] ButtonClick	User clicked the browser’s home button, back button, forward button, or reload button.

1297118695677 7-2-2011 17:44:55 S42 P14 T2 intask LoadCap http://www.cnn.com/ http://www.cnn.com/
--

Fig. 7. Example line of a log file

After the eight fixed fields, the remaining logging information on each line depends on the event. Many events record the URL of the page that was in focus when the event occurred, and/or the URL of the page that is the target of the event. Events also include a final field that contains the most recently focused URL (which may be the same as the other URL listed).

Fig. 8 shows a portion of a log file that illustrates logging of the interactions and answers given by a user for a pre-task questionnaire. Note that in Fig. 8 parts of the log file have been omitted for space. When the pre-task questions are displayed, information is logged about the task description (e.g. at time 7110 in Fig. 8) and the maximum number of answers for this task (7113). At time 7732, the user finished entering answers to the pre-task questions and clicked “OK” to move on to start the task. The answers provided to questions q1, q2, and q3 are then logged (7736, 7739, 7741) and at time 7893 the log switches to logging events “intask”.

TIME	EVENT	DETAILS
7110	pretask	TaskDescription What is being done to help
7113	pretask	TaskMaxAnswers 3 null
7732	pretask	onAccept event -- ok pressed for Task #2
7736	pretask	q1 4 null
7739	pretask	q2 3 null
7741	pretask	q3 i plan to use search engines and look
7893	intask	Focusabout:blank about:blank

Fig. 8. Example pretask section of a log file

Fig. 9 shows a more extensive example of a log file, again with portions omitted for space. This section shows that at time 5677, the user loaded a Google search results page and then right-clicked on a link (0323). The right-click was to load the link in a new tab (1798). When a new tab is opened, the HCI Browser logs a summary of all the open windows and tabs (1821, 1824, 1828). At the time the summary was recorded (1828), tab #1 had not yet loaded the new page, so it is listed as url=about:blank, but the load event for the www.cdc.gov page is next in the log (2865). Notice that the www.cdc.gov page is loaded in the new tab but the user does not focus away from the Google search results page at this point. The user is still interacting with the Google search results and issues several scroll events to browse the results (5384, 6539, 7174). The user then right-clicks and loads two other pages from the search results (a page from aspe.hhs.gov and an en.wikipedia.org page), again without navigating away from the Google search results page. Finally, at 3823, the user selects a different tab, switching the focus to the www.cdc.gov page (3844). The user then scrolls on this page (5470, 6271), and left-clicks a link on the cdc.gov site about “keystrategies” (9875).

6 Implementation

The HCI Browser is written as a Firefox Web browser extension, packaged as an XPI file. The browser interface components are written in the XUL user-interface description language and the program logic is implemented in Javascript. It is outside the scope of this paper to provide details about Firefox extension development, but there are many good tutorials (e.g. [2]). The concepts implemented in the open-source Lemur Query Log Toolbar are also helpful in understanding Firefox toolbar creation and event logging and the HCI Browser has been influenced and informed by many of its concepts [11].

The HCI Browser monitors three main types of events in the browser: 1) events in the browser user interface (e.g. back/reload button, typing in the address bar, menu activity), 2) events that deal with windows, tabs, and page loading, and 3) events that occur on the currently focused page (e.g. mouse clicks). In order to monitor the mouse events on the Web pages, when each page is loaded, a “mousedown” event handler is added to every link found on the page.

Sometimes a single Web page will cause several underlying Firefox “load” or “focus” events to occur. The HCI Browser employs filters so that if multiple, sequential LoadCap or Focus events are detected for the same URL, only the first will be written to the log file.

TIME	EVENT	DETAILS
5677	LoadCap	http://www.google.com/search?hl=en&source=hp&b
0323	RClick	http://www.google.com/url?sa=t&source=web&cd=1
1798	AddTab	about:blank http://www.google.com/search?hl=en
1821	Info	Currently open windows and tabs are logged bel
1824	Info	Window=0, tab=0, url= http://www.google.com/sea
1828	Info	Window=0, tab=1, url=about:blank http://www
2865	LoadCap	http://www.cdc.gov/HealthyYouth/obesity/ ht
5384	Scroll	http://www.google.com/search?hl=en&source=hp&b
6539	Scroll	http://www.google.com/search?hl=en&source=hp&b
7174	Scroll	http://www.google.com/search?hl=en&source=hp&b
0043	RClick	http://aspe.hhs.gov/health/reports/child_obesi
0996	AddTab	about:blank http://www.google.com/search?hl=en
1009	Info	Currently open windows and tabs are logged bel
1011	Info	Window=0, tab=0, url= http://www.google.com/sea
1032	Info	Window=0, tab=1, url= http://www.cdc.gov/Health
1050	Info	Window=0, tab=2, url=about:blank http://www
1946	LoadCap	http://aspe.hhs.gov/health/reports/child_obesi
3033	RClick	http://www.google.com/url?sa=t&source=web&cd=1
4101	AddTab	about:blank http://www.google.com/search?hl=en
5004	Scroll	http://www.google.com/search?hl=en&source=hp&b
6256	LoadCap	http://en.wikipedia.org/wiki/Childhood_obesity
3823	SetTab	http://www.cdc.gov/HealthyYouth/obesity/ ht
3844	Focus	http://www.cdc.gov/HealthyYouth/obesity/ ht
5470	Scroll	http://www.cdc.gov/HealthyYouth/obesity/ ht
6271	Scroll	http://www.cdc.gov/HealthyYouth/obesity/ ht
9875	LClick	http://www.cdc.gov/HealthyYouth/keystrategies/

Fig. 9. Example log file. Parts of the file have been omitted and the time stamp has been shorted in order to fit the space.

7 Summary

The HCI Browser is a tool to help support data collection and administration of studies of Web information seeking. It is released as “alpha” level, open-source software. More details, including information about downloading and installing the HCI Browser are at: <http://ils.unc.edu/hcibrowser>

References

1. Bierig, R., Gwizdka, J., Cole, M.J.: A user-centered experiment and logging framework for interactive information. In: Proceedings of the SIGIR 2009 Workshop on Understanding the User: Logging and Interpreting User Interactions in Information Search and Retrieval, Boston, MA, July 23, pp. 8–11 (2009)
2. Bishop, J.: Born Geek Firefox Toolbar Tutorial, <http://www.borngeek.com/firefox/toolbar-tutorial/> (retrieved February 10, 2011)
3. Capra, R.: HCI Browser: A Tool for Studying Web Search Behavior. Poster Presentation at the 73rd American Society for Information Science and Technology Annual Meeting (2010)

4. Capra, R.: The HCI Browser Tool for Studying Web Search Behavior. In: Proceedings of the Third Workshop on Human-Computer Interaction and Information Retrieval, pp. 38–41 (2009)
5. Capra, R.: HCI Browser: A Tool for Studying Web Search Behavior. In: Workshop on Understanding the User at ACM SIGIR (2009)
6. Capra, R.: Studying Elapsed Time and Task Factors in Re-Finding Electronic Information. In: Workshop on Personal Information Management at ACM CHI (2008)
7. Claypool, M., Le, P., Wased, M., Brown, D.: Implicit Interest Indicators. In: Proceedings of the 6th International Conference on Intelligent User Interfaces (2001)
8. Feild, H., Allan, J., Jones, R.: Predicting Searcher Frustration. In: The Proceedings of the 33rd International ACM Conference on Research and Development in Information Retrieval (SIGIR 2010), Geneva, Switzerland, July 19–23 (2010)
9. Kaasten, S. and Greenberg, S.: URL Tracer: URL Tracing Software for Internet Explorer, <http://grouplab.cpsc.ucalgary.ca/cookbook/index.php/Utilities/URLTracer> (retrieved on February 7, 2011)
10. Keller, M., Hawkey, K., Inkpen, K., Watters, C.: Challenges of Capturing Natural Web-Based User Behaviors. *International Journal of Human-Computer Interaction* 24(4), 385–409 (2008)
11. Lemur Query Log Toolbar, <http://www.lemurproject.org/querylogtoolbar>
12. Mozilla Developer Network Events, <https://developer.mozilla.org/en/XUL/Events> (retrieved February 10, 2011)
13. Reeder, R.W., Pirolli, P., Card, S.K.: WebEyeMapper and WebLogger: Tools for Analyzing Eye Tracking Data Collected in Web-Use Studies. In: CHI 2001 Extended Abstracts (2001)
14. Turnbull, D. Webtacker: A Tool for Understanding Web Use, <http://www.ischool.utexas.edu/~donturn/research/webtracker/index.html> (retrieved on February 7, 2011)