Big Brother: A Drop-In Website Interaction Logging Service

Harrisen Scells
h.scells@uq.net.au
The University of Queensland
Brisbane, Australia

Jimmy
jimmy@ubaya.ac.id
University of Surabaya
Surabaya, Indonesia

Guido Zuccon
g.zuccons@uq.edu.au
The University of Queensland
Brisbane, Australia

ABSTRACT

Fine-grained logging of interactions in user studies is important for studying user behaviour, among other reasons. However, in many research scenarios, the way interactions are logged is usually tied to a monolithic system. We present a generic, application-independent service for logging interactions in web-pages, specifically targetting user studies. Our service, Big Brother, can be dropped-in to existing user interfaces with almost no configuration required by researchers. Big Brother has already been used in several user studies to record interactions in a number of user study research scenarios, such as lab-based and crowdsourcing environments. We further demonstrate the ability for Big Brother to scale to very large user studies through benchmarking experiments. Big Brother also provides a number of additional tools for visualising and analysing interactions.

Big Brother significantly lowers the barrier to entry for logging user interactions by providing a minimal but powerful, no configuration necessary, service for researchers and practitioners of user studies that can scale to thousands of concurrent sessions. We have made the source code and releases for Big Brother available for download at https://github.com/hscells/bigbro.

KEYWORDS

user studies, interaction logging

ACM Reference Format:
Harrisen Scells, Jimmy, and Guido Zuccon. 2021. *Big Brother*: A Drop-In Website Interaction Logging Service. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15,2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3462781

1 PROBLEM, TARGET USERS & IMPORTANCE

Recording interactions in a user study setting is extremely important for understanding user behaviour, among other reasons. However, writing the code to implement the recording of interactions can be tedious and a burden on those who wish to run user studies. Indeed, there is a considerable amount of software infrastructure to be considered when logging user interactions: How to efficiently process streams of interactions from multiple users/sessions? How to store the logs? How to deal with all the types of interactions a user might make? When we encountered these questions, it was clear that a *standard*, *simple*, and *drop-in* service was required. The requirement for the service to passively log interactions once loaded into a web page also provides a generic way to capture events from users no matter the framework or setting that the user study is run. We present this service to the wider IR community as an Open Source tool, dubbed *Big Brother*. An overview of the Big Brother architecture is presented in Figure 1.

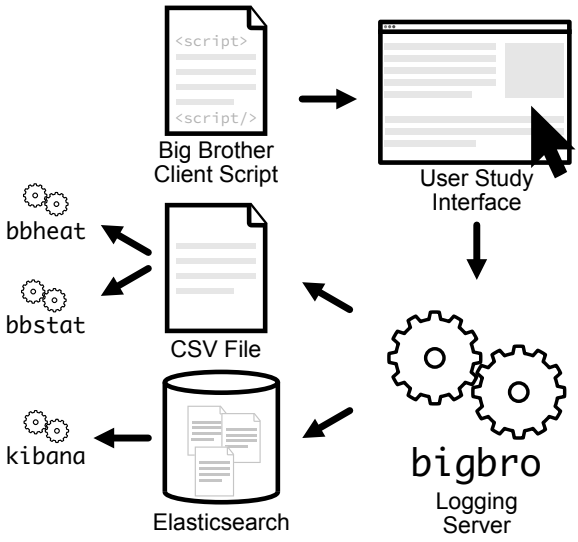

Figure 1: High-level overview of the system architecture.

Fine-grained interaction logs enable researchers to more deeply understand how users behave in their systems. Tracking and analysing how users behave is important for a number of reasons:

Evaluation. While users self-reported measurements can provide a weak indication of the effort required to complete tasks, tracking user interactions can provide a stronger source of evidence for how much effort, or how challenging a user found a task. For example, by logging user interactions with Big Brother, Cross et al. [4] found that user reported effort differed from the number of interactions between two systems.

User Modelling. Recording how users interact with systems enables a deeper understanding of user behaviour, and can lead to models that better reflect the needs of users. In Information Retrieval tasks, White [12] notes that modelling user behaviour permits "better understanding the search process, estimating satisfaction, identifying connections between queries and/or URLs, and learning which results are relevant."

Multi-Modal Interactions. Often it is desirable to analyse behavioural patterns between how users interact with the keyboard and mouse and other physiological interactions, such as e facial

SIGIR'21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462781

expressions [1]. For example, using Big Brother, Jimmy et al. [7] were able to better understand what users paid attention to by combining interaction data like clicks and mouse movements with eye-tracking.

Malicious Users. By tracking how frequently a user interacts with elements on a page it is possible to identify behavioural patterns that may correspond with malicious intent [5].

2 OPERATION OF THE SYSTEM

Big Brother has two main components: a high-throughput server and a JavaScript client. The client listens to events that occur in a web browser, and the server ingests them in real-time.

2.1 Client

The Big Brother client has been written to provide maximum functionality with minimal setup and configuration. The smallest amount of code to have Big Brother listen to events is presented in Listing 1. By default, Big Brother is configured to listen to all events on all elements on a page. This client has been put to the test on multiple user studies with highly complex interfaces [4, 7, 8]. Practitioners of these studies and users of their systems noticed no slowdown of their study as a result of using Big Brother.

Listing 1: Minimal example of code required to initialise `bigbro` on the client-side. Note the `session_id` which should already be initialised elsewhere.

```
<script src="bigbro.js"></script>
<script type="text/javascript">
BigBro.init(session_id, "localhost:1984");
</script>
```

Often, however, the practitioner of a user study may only be interested in a subset of events. Listing 2 presents the JavaScript code required on the client-side to restrict Big Brother to listen to two events: `mousemove` and `onload`, which will record the coordinates of the users' mouse, and the time the session began respectively.

Listing 2: Initialising `bigbro` to listen on only certain global events. HTML code removed for brevity.

```
BigBro.init(session_id,
            "localhost:1984",
            ["mousemove", "onload"]);
```

Sometimes, the practitioner of a user study may wish to capture custom interactions or events from their user. For example, logging a custom event for when a user clicks a specific button on the page. Listing 3 presents the JavaScript code required on the client-side to wire the processing of a custom event to Big Brother's exposed custom logging functionality.

Listing 3: Wiring up `bigbro` to listen to click events and to log a custom event.

```
let bb = BigBro.init(session_id,
                     "localhost:1984");
let w = window;
w.addEventListener("click", function (e) {
    bb.log(e, "custom_event");
})
```

Component	Description of Component
Target	The HTML element that has caused the event to trigger.
Name	The `name` attribute of the HTML element in Target.
ID	The `id` attribute of the HTML element in Target
Method	The name of the method that caused the event to trigger.
Location	The web-page location on the server (URL, with query string and anchors).
Comment	Any additional custom information that may be useful to interpreting the event.
X	The x-position within the web browser that the event occurred at.
Y	The y-position within the web browser that the event occurred at.
ScreenWidth	The width of the web browser the event occurred within.
ScreenHeight	The height of the web browser the event occurred within.
Time	The time the event happened.
Actor	An identifier that can be used to refer to the user that caused the event, e.g., session ID.

Table 1: The components of an event.

Finally, it is possible to record the screen using Big Brother. Listing 4 presents the minimum JavaScript code required to start a live capture of the browser window a user is looking at. Because of browser security, screen capture cannot be started automatically, and must be initiated by clicking an element (as seen in the argument provided to the function in Listing 4). Captures are saved as a series of images. The time between each capture can be configured, and captures can be taken in response to events. The images can be stitched together into a video later, or can be used to highlight specific interactions from the logs.

Listing 4: Wiring up `bigbro` to listen to click events and to log a custom event.

```
let bb = BigBro.init(session_id,
                     "localhost:1984");
bb.startCaptureOnClick("capture");
```

The Big Brother client has been specifically designed to require very minimal configuration and setup. Likewise, the server is designed to be run independently of components such as the web server a user study is hosted on.

2.2 Server

The server is written in the Go programming language. This provides a number of advantages, primarily the ability for the server code to be compiled into a high-performance single-binary file for multiple platforms (macOS, Windows, Linux, etc.). The server can be configured using command-line options. It can also be integrated into existing web servers as library code, so long as those servers are written in Go.

The server processes interactions from users as *events*. The components that comprise an event are summarised in Table 1. An event captures information about the HTML elements that were interacted with (**Target**, **Name**, **ID**), how the event was triggered (**Method**), what the URL was on the page that the event occurred (**Location**), additional custom comments (**Comment**), positional information to "replay" interactions (**X**, **Y**, **ScreenWidth**, **ScreenHeight**), the time that the event occurred (**Time**), and the user that caused the event (**Actor**).

Events are streamed from the user's web browser to a centralised server over websockets. This enables the real-time ingestion of interactions. Currently, Big Brother can output logs to a local csv file as well as directly indexing them to Elasticsearch. A small, real example of logs captured during a user study is presented in Listing 5. The researcher who ran this study decided to output their interaction logs directly to a csv file. The ordering of components in these events is how they are produced by `bigbro` . Listing 6 shows the ordering of components in the csv file output. These logs show the interactions of two user sessions from a human intelligence task (HIT) on a popular crowdsourcing platform. This slice of interaction logs begins when the `A1` user starts a HIT, and ends when they finish the HIT. At the same time, another user (`A2`) is in the middle of another HIT. The researcher running this user study records when their users click on a multiple select dropdown. The very final line in the log, when user `A1` finishes the HIT, also logs the recorded answers to the questionnaire using the Comment component: `T|T|T|T|T`.

In addition to allowing the researcher to see in real-time how users are interacting with their experiments (and potentially throw away sessions from users who completed the study too quickly), it provides a second way to record responses that users make to questionnaires.

2.3 Throughput Benchmark

We demonstrate the ability of Big Brother to scale and accommodate very large user studies by providing a throughput benchmark of logging user interactions directly to a csv file. To make the benchmark more realistic, we simulate the time it takes to process an event by encoding and decoding the event, and populate the event with random data. Each event has Target, Method, Location, X, Y, ScreenHeight, ScreenWidth, Time and Actor components. To benchmark the code, we use the standard benchmarking framework included in the Go language (as this is the language that the server is written in). In this framework, a loop containing the process described earlier is run N times until it takes a second for the benchmark to complete. We report the total number of events processed, the average time it takes to process one event in microseconds, and the throughput in MB/s. We run the benchmark ten times and report both each run and the overall statistics. The benchmarks were run on a 2.7GHz Intel Core i7 8-core 2018 MacBook Pro with 16GB of LPDDR3 RAM and solid-state flash storage. Table 2 presents the results of the benchmark. On average, Big Brother can process approximately 70,000 events per second, with each event processed in approximately 26μs, and write the interactions to a file at a speed of approximately 5MB/s. In practice, while we never came close to this amount of interactions per second, `bigbro` did remain resilient and experienced no faults over the course of a number of studies.

Processed Events	μs/Event	Throughput (MB/s)
79,026	22.21	5.58
71,455	29.51	4.27
37,600	32.79	3.87
84,356	31.22	4.07
60,418	19.31	6.53
82,922	23.70	5.36
72,714	30.97	4.10
83,862	27.30	4.61
86,094	24.86	5.07
51,250	24.37	5.21
70,969.70±16,326.29	26.60±0.27	4.87±0.34

Table 2: Benchmark results of logging events. The last row indicates average across the runs, with standard deviation. The processed events column indicates the number of events processed in one second.

2.4 Other Tools within Big Brother

The Big Brother family also includes other tools for visualising and manipulating data produced by the `bigbro` tool. These include `bbheat` and `bbstat` , which are compatible with CSV formatted logs produced by `bigbro` . If Elasticsearch is used as the storage medium for logs, it is possible to use the infrastructure that has been developed for it, including Kibana, a data visualization dashboard. Note that it is possible to use the Big Brother tools and the Elasticsearch tools by allowing `bigbro` to write to a CSV file and ingesting that file into Elasticsearch using software such as Logstash. For details on how to use these tools, please refer to the GitHub repository.

2.5 `bbheat`

Often it is useful to be able to visualise the interactions of users over user interfaces. For this, the `bbheat` tool can be used. This tool allows one to produce a static or animated heat map of mouse movements. The tool has several options for filtering interactions, for example by actor, time, and browser window size. An example of one of the heat maps produced by this tool is presented in Figure 2. Using the screen recording functionality of Big Brother, it becomes trivial to visualise how different users interact with a system, or how changes in interface design can impact user behaviour.

2.6 `bbstat`

Analysing log data can be tedious, especially processing the data for analysis. The `bbstat` tool provides common functionality for processing interaction logs for later analysis. The functionality provided by the `bbstat` tool include filtering: interaction logs can be split into multiple files based on actor, time, or comment, and aggregation: production of reports such as how long a user spent on different pages or how many times certain elements in a page were clicked.

2.7 Kibana

Kibana is a product that is developed by the same company that built Elasticsearch. It provides tools for visualising and manipulating data. Kibana has been developed with time series logs in mind. A deeper description of Kibana is not provided here for space reasons, but it can be a powerful alternative to the tools provided by Big Brother.

Listing 5: Example log file capturing two user sessions at the same time. Note that there were far more interactions captured during the study and that this is a small contrived slice of the data. Much of the information from the URL has been redacted.

```
19-01-03 06:12:45,A1,hit_start,,,,https://example.com?assignmentId=XYZ123&hitId=HIT1,0,0,1319,646,
19-01-03 06:12:47,A1,click,SELECT,q1,,https://example.com?assignmentId=XYZ123&hitId=HIT1,1079,299,1319,646,
19-01-03 06:12:48,A1,click,SELECT,q1,,https://example.com?assignmentId=XYZ123&hitId=HIT1,0,0,1319,646,
19-01-03 06:12:49,A1,click,SELECT,q2,,https://example.com?assignmentId=XYZ123&hitId=HIT1,1067,328,1319,646,
19-01-03 06:12:46,A2,click,SELECT,q4,,https://example.com?assignmentId=ABC789&hitId=HIT2,806,233,1063,306,
19-01-03 06:12:46,A2,click,SELECT,q4,,https://example.com?assignmentId=ABC789&hitId=HIT2,0,0,1063,306,
19-01-03 06:12:50,A1,click,SELECT,q2,,https://example.com?assignmentId=XYZ123&hitId=HIT1,0,0,1319,646,
19-01-03 06:12:50,A1,click,SELECT,q3,,https://example.com?assignmentId=XYZ123&hitId=HIT1,1067,354,1319,646,
19-01-03 06:12:51,A1,click,SELECT,q3,,https://example.com?assignmentId=XYZ123&hitId=HIT1,0,0,1319,646,
19-01-03 06:12:49,A2,click,SELECT,q5,,https://example.com?assignmentId=ABC789&hitId=HIT2,779,260,1063,306,
19-01-03 06:12:52,A1,click,SELECT,q4,,https://example.com?assignmentId=XYZ123&hitId=HIT1,1069,390,1319,646,
19-01-03 06:12:50,A2,click,SELECT,q5,,https://example.com?assignmentId=ABC789&hitId=HIT2,0,0,1063,306,
19-01-03 06:12:53,A1,click,SELECT,q4,,https://example.com?assignmentId=XYZ123&hitId=HIT1,0,0,1319,646,
19-01-03 06:12:54,A1,click,SELECT,q5,,https://example.com?assignmentId=XYZ123&hitId=HIT1,1074,424,1319,646,
19-01-03 06:12:55,A1,click,SELECT,q5,,https://example.com?assignmentId=XYZ123&hitId=HIT1,0,0,1319,646,
19-01-03 06:12:56,A1,hit_end,INPUT,,submit,https://example.com?assignmentId=XYZ123&hitId=HIT1,659,617,1319,646,T|T|T|T|T
```

Listing 6: Order of event components in Listing 5

```
Time,Actor,Method,Target,Name,ID,Location,X,Y,ScreenWidth,ScreenHeight,Comment
```

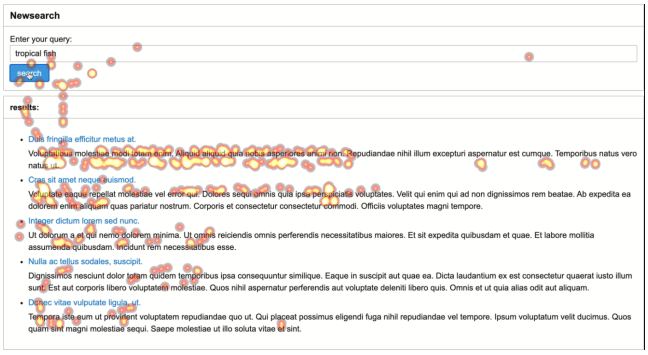

Figure 2: Heat map of mouse interactions overlaid on top of an image of a search interface captured using the screen recording functionality of Big Brother.

3 COMPARISON WITH EXISTING SYSTEMS

The idea to capture interactions in user study scenarios is not new. For example, Zuccon et al. [13] proposed a system for evaluating Information Retrieval systems using interactions from crowdsourced workers. They observe interactions such as the length of sessions, the number of elements clicked, and the number of items examined. However, they do not measure fine-grained interactions such as mouse movements, and these measurements are built into the underlying interface, unlike Big Brother which can log interactions independently of the underlying interface. Bierig et al. [3] proposed an end-to-end system for designing user studies and logging multi-modal interactions within those studies. The downside to this system is that it may not provide the flexibility afforded by bespoke user study systems. Big Brother is able to be integrated easily into highly bespoke web-based systems. Atterer and Albrecht [2] proposed a similar interaction logging system to ours by using AJAX requests instead of web sockets. However, their events contain fewer components than ours, and it is unlikely that this older technology would scale for large user studies of today. More recently, Maxwell and Hauff [11] have developed fine-grained logging infrastructure for web-based experiments that use similar modern technology to ours. While the architecture of their system may be similar to ours, we believe that the barrier to entry for logging interactions is significantly lower using Big Brother and the fact that Big Brother has been used already in a number of user studies (lab-based and embedded inside crowdsourcing services) demonstrates its robustness in different user study scenarios. We further contrast Big Brother to their LogUI system by demonstrating the additional ability to record the screen of users and the associated tools for visualising and manipulating interactions.

Lettner and Holzmann [10] and Kokemore and Hutter [9] both proposed toolkits for automatically logging user interactions in mobile applications. Their frameworks differ from ours as they target mobile settings (e.g., Android or iOS apps) instead of web-based interfaces. Note however that Big Brother is compatible with touch events in a mobile browser. In a similar vein, Jeong et al. [6] proposed a framework for visualising how users transition from one mobile application screen to the next. This system captures a screenshot each time the user transitions to a new interface in the app. This functionality is exposed in Big Brother, but this system can provide a transition graph visualisation, which we do not.

4 IMPACT OF BIG BROTHER

The Big Brother logging system has been used already in several user studies and has been 'battle tested'. We demonstrate the ability of the `bigbro` server to scale to tens of thousands of interactions per second. We have developed the Big Brother ecosystem to allow researchers to integrate interaction logging into any web-based system. Furthermore, we have designed the entire system to be extremely easy to use, while providing highly sophisticated functionality, such as screen recording. These attributes, we believe, will significantly lower the barrier to entry for capturing user interactions in many research settings.

Acknowledgements. We would like to thank Anton van der Vegt and Sebastian Cross for their use of Big Brother in their user studies and Ahmed Mourad for proofreading a draft of this paper. Dr Guido Zuccon is the recipient of an Australian Research Council DECRA Research Fellowship (DE180101579) and Google Faculty Award. This research is supported by the Grain Research and Development Corporation (GRDC), project AgAsk (UOQ2003-009RTX).

REFERENCES

[1] Ioannis Arapakis, Joemon M Jose, and Philip D Gray. 2008. Affective Feedback: An Investigation into the Role of Emotions in the Information Seeking Process. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 8.

[2] Richard Atterer and Albrecht Schmidt. 2007. Tracking the interaction of users with AJAX applications for usability testing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1347–1350.

[3] Ralf Bierig, Jacek Gwizdka, and Michael Cole. 2009. A User-Centered Experiment and Logging Framework for Interactive Information Retrieval. *Understanding the User-Logging and Interpreting User Interactions in Information Search and Retrieval (UIIR-2009)* (2009), 8.

[4] Sebastian Cross, Ahmed Mourad, Guido Zuccon, and Bevan Koopman. 2021. Search Engines vs. Symptom Checkers: A Comparison of Their Effectiveness for Online Health Advice. In *Proceedings of the 2021 Web Conference*. 11.

[5] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. 2015. Understanding Malicious Behavior in Crowdsourcing Platforms: The Case of Online Surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, Seoul Republic of Korea, 1631–1640.

[6] JongWook Jeong, NeungHoe Kim, and Hoh Peter In. 2020. GUI information-based interaction logging and visualization for asynchronous usability testing. *Expert Systems with Applications* 151 (2020), 113289.

[7] Jimmy, Guido Zuccon, Gianluca Demartini, and Bevan Koopman. 2020. Health Cards to Assist Decision Making in Consumer Health Search. In *AMIA Annual Symposium Proceedings*, Vol. 2019. 1091–1100.

[8] Jimmy Jimmy, Guido Zuccon, Bevan Koopman, and Gianluca Demartini. 2019. Health Cards for Consumer Health Search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 35–44.

[9] Ilka Kokemor and Hans-Peter Hutter. 2016. Aspect-Oriented Approach for User Interaction Logging of iOS Applications. In *International Conference of Design, User Experience, and Usability*. Springer, 45–56.

[10] Florian Lettner and Clemens Holzmann. 2012. Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*. 118–127.

[11] David Maxwell and Claudia Hauff. 2021. LogUI: Contemporary Logging Infrastructure for Web-Based Experiments. In *Proceedings of the 43rd European Conference on Information Retrieval*.

[12] Ryen W. White. 2016. *Interactions with Search Systems*. Cambridge University Press.

[13] Guido Zuccon, Teerapong Leelanupab, Stewart Whiting, Emine Yilmaz, Joemon M. Jose, and Leif Azzopardi. 2013. Crowdsourcing Interactions: Using Crowdsourcing for Evaluating Interactive Information Retrieval Systems. *Information retrieval* 16, 2 (2013), 267–305.