YASFIIRE: Yet Another System for IIR Evaluation

Xing Wei
School of Information
University of Texas, Austin
weixing1985@gmail.com

Yinglong Zhang
School of Information
University of Texas, Austin
ylZhang@utexas.edu

Jacek Gwizdka
School of Information
University of Texas, Austin
iiix2014@gwizdka.com

ABSTRACT
We present a system that supports Interactive Information Retrieval user studies on the Web. Our system provides support for user and task management, for processing web-based task specific interfaces and for Web-event logging. It also offers functionality useful to IIR studies that capture eye-movement on Web page elements. The system complements logging functionality offered by a typical usability/eye-tracking software packages and is designed to act in concert with such software.

Categories and Subject Descriptors
H3.3 Information Search and Retrieval: Search process; H.5.2 User Interfaces: Evaluation/methodology.

General Terms
Human Factors.

Keywords
Interactive information retrieval; evaluation; experiment support.

1. INTRODUCTION
In contrast to the traditional Cranfield-style IR evaluations, Interactive Information Retrieval (IIR) evaluations offer numerous challenges due to lack of standardized collections, research protocols and measures. The situation has improved in recent years with several proposals that targeted the lack of standardization (e.g., [5][6]). A number of frameworks and systems for IIR evaluation have also been proposed (e.g., [2][8]).

Why propose another system for IIR experiments on the Web? We created our system to complement logging functionality offered by typical usability/eye-tracking software packages and to act in concert with such software; to enable on-the-fly processing and modification of external web-based interfaces; to capture coordinates of web interface elements in support of eye-tracking data analysis; and to provide essential IIR experiment management support. Accordingly, our system provides support for user and task management, for processing web-based task specific interfaces and for logging Web-events. It also offers functionality useful to IIR studies that need to capture eye-movement on Web page elements.

2. RELATED WORK
One of the first systems created expressly for IIR experiments was WiIRE (Web Interactive IR Experiments tool) [9]. The tool was designed to support creating and conducting experiments with Web-based user interfaces. WiIRE enabled experiment management and included delivery of questionnaires, tasks, task interfaces and other essential experiment components. WiIRE logged questionnaire responses and task completion times, but it did not support fine-grain user interaction logging.

Framework presented in [2] was focused on capturing and integrating user interaction from multiple loggers, on support for experiment management (user and task management), and on storage and representation of data recorded from multiple sources as well as on data analysis. The framework supported extensive logging through the use of third-party loggers. Their use, however, required a not-necessarily-trivial customization at the software and data representation levels.

SCAMP [8] was a system designed on top of PuppyIR[2] framework. The system supported creation of typical experimental designs. Noteworthy was its ease of use, mainly due to no need for customization or coding. SCAMP came also with some limitations. Firstly, this system only supported within-subject experimental design, hence it could not be used in some more complex experimental designs, such block design and mixed design. Secondly, SCAMP did not record participants' search actions. Lastly, if a part of an experiment failed to be finished, SCAMP didn't allow participants to resume completion of the unfinished tasks.

Eye-tracking has received a extensive attention in IIR studies in the last decade [7]. The work often focused on recording and analyzing eye fixation patterns on ranked search results pages (e.g., [4]). However, eye-tracker loggers provide a rather weak support for automated capture of eye fixations on dynamic areas of interest, such as search box or individual search results. A few researchers created their own solutions for automated capture of AOIs. For example, Tran and Fuhr's AOILog system [10] tracks position, visibility and size of users interface objects, so that user's gaze on this objects can be captured and analyzed. However, AOILog system is designed in a Java-based framework and its interface element capture ability is limited to Java Swing GUI objects. Thus, it only supports IIR interfaces written in Java. Moreover, AOILog needs to be recompiled with an IIR interface before it can be used. Unlike AOILog, our system, YASFIIRE uses JavaScript to automate AOI capture. JavaScript can be run inside most Web browsers without further modification, thus, YASFIIRE supports more widely available search task interfaces implemented in HTML-related technologies. Others who employed a similar approach include WebGazeAnalyzer project [1], in which researchers captured coordinates of web page DOM (document-object model) structure. That project, however, focused only on eye gaze capture on web pages. It did not address capture of other interactions nor experiment management. In another project, Buscher et al. [3] used JavaScript inside web pages to record web page elements of SERPs and log mouse interaction. Although, their approach is similar to ours, their main project goal was to capture user search interactions at a large scale and not using eye-tracking.

IIiX '14, Aug 26-29 2014, Regensburg, Germany
ACM 978-1-4503-2976-7/14/08.
http://dx.doi.org/10.1145/2637002.2637051

YASFIIRE shares some similarities with each type of the described systems. Yet, it integrates the different functionalities in a new way. YASFIIRE provides basic functionality in support of managing IIR research experiments. It records participants' webpage interactions such as loading, scrolling, and closing of pages. It records coordinates of Web-based interface elements needed for eye movement data analysis. It allows for modification of third-party Web-based interfaces, thus enabling a number of different IIR evaluation scenarios.

3. SYSTEM ARCHITECHTURE

We present our system (Figure 1) by referring to the general evaluation framework proposed by Hall & Toms in [5]. Our system provides support for functionalities identified by them as belonging to three core components: Research Manager, Experiment Manager, and Data Extractor ([5], Fig 1.).

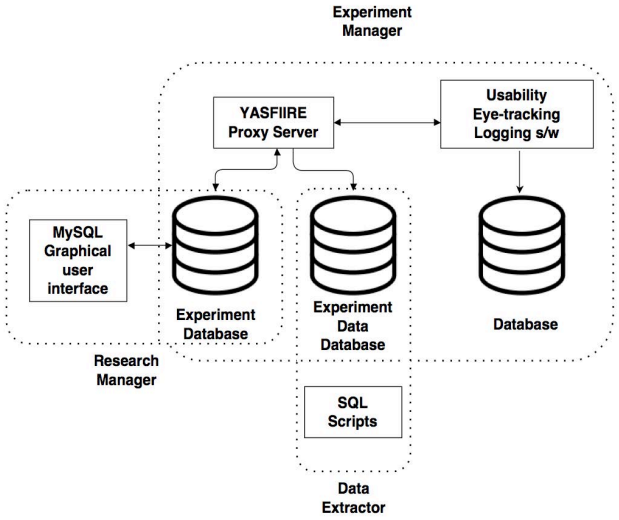

Figure 1. System architecture: the main blocks.

The **Research Manager** serves to define users, tasks, task rotations along with their assignment to users. Our philosophy in creating YASFIIRE was to take advantage of existing software packages and not to provide a turn-key system. This approach allows for a greater flexibility in the types of supported experiments, but it comes at a cost of a higher expertise required from researches using the system. For example, the experiment user and task information is entered using a standard MySQL database GUI interface. A researcher uses this interface to access and modify experiment database. This database plays an important role in YASFIIRE as it stores experimental design and maintains current status of experiment execution and tracks the lifecycle of an ongoing experiment. This functionality allows for resuming experiment at the most recent task, in case some system components fail during an experiment.

The **Experiment Manager** functionality is distributed between the YASFIIRE proxy server – the heart of our system (described in detail later – see Figure 2), and a third-party usability/eye-tracking logging software. The proxy server uses experiment database to deliver tasks along with associated search task interfaces. It controls task rotations, its association with users, and it knows which interface to present and, if applicable, how to modify it before displaying it. The timing of the delivery is controlled by a participant through interaction with the usability/eye-tracking logging software. Researcher loads into this software a list of identifiers (such as URLs) that define types of experiment interfaces presented to a participant. The types of experiment interfaces we currently support are: task description display interface, questionnaires (e.g., a pre-/post-questionnaires) and task specific interfaces (such as, search engine interfaces). The usability/eye-tracking logging software does not control task order, nor does it control which task is being conducted – both are under control of the proxy server.

In the spirit of our philosophy, our system does not offer logging of lower granularity user interactions. Instead, it relies on this functionality being provided by third party usability/eye-tracking logging software, which is used to control lower granularity data logging (e.g., mouse clicks, key presses, screen contents, eye-tracking data).

The **Data Extractor** takes the experiment data and saves it in a tabular format common to many statistical packages (e.g., SPSS) or languages (R). Part of its functionality is to load the data exported from a usability/eye-tracking logger to a database and link it with the experiment data.

4. SYSTEM FUNCTIONALITY

IIR experiments can be conducted using own task user interfaces or external task interfaces. Our system supports building the experiment environment on top of pre-existing task specific interfaces located on external Websites. In particular, our focus is on supporting task specific interfaces located on remote servers that experimenters do not control.

The core component of the system is the Request/Response Processor written in Node.js. The R/R processor serves as an intermediate component that monitors and modifies HTTP requests and responses between the local computer and the remote server (Figure 2).

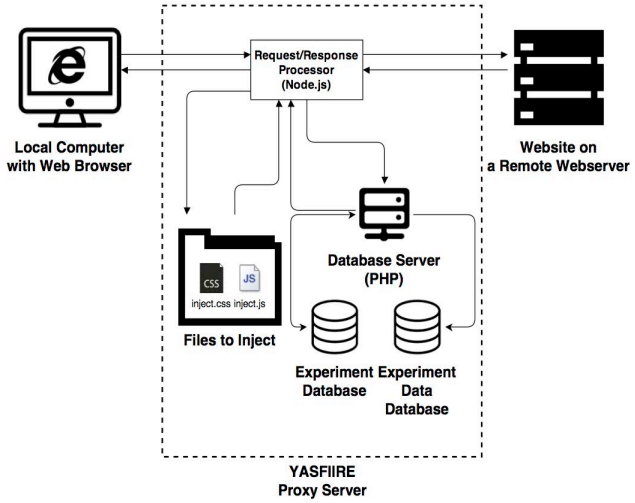

Figure 2. More detailed system architecture showing the process of loading external web-based interfaces.

4.1 Experiment server

Our proxy is as a full-featured server with its own database (Experiment Database) that controls entire IR experiment lifecycle (user and task management) and with a "write-only" database (Experiment-Data Database) that stores user responses, webpage interactions (such as, loading, scrolling, and closing of pages), task completion times, and coordinates of selected web page elements, in support of eye-tracking data analysis.

4.2 Reconstruction of Web page interfaces

One goal of our system is to support IIR experiments that use task interfaces located on remote servers. Our system modifies a number of elements on a Web page including its appearance and interaction mode. This is achieved by having our proxy server inject JavaScript files and CSS files into Web pages that are requested from a remote server (Figure 3). This technique allows us to provide several types of functionality, we give two examples below.

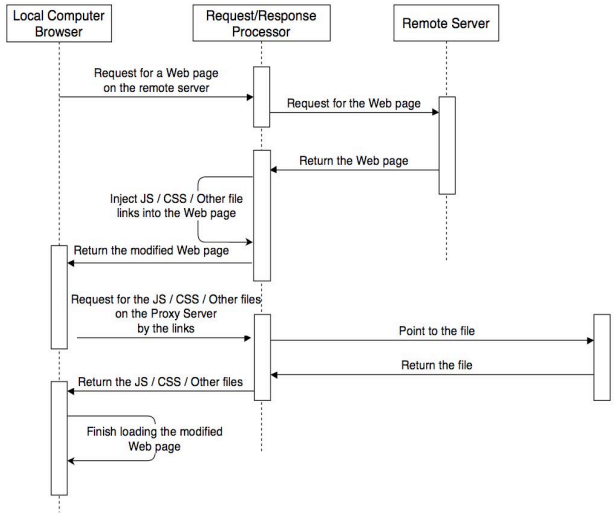

Figure 3. The process of loading a Web page from a remote server.

4.2.1 Extended bookmarks and notes on web pages

In one of our IIR experiments, a user needs to be able to collect and annotate web pages of interest, as well as to delete and modify the added annotations. This functionality extends the web browser bookmarks. Our extended bookmarks and their interface controls are displayed in the left-sidebar, next to a task interface. They are integrated into one web page together with a web-based task interface from an external website (***Figure 4***). The bookmarks and annotations are stored in the experiment data database. This functionality is provided by the Request/Response Processor together with PHP Database Server (Figure 2).

4.2.2 Support for eye-tracking data analysis

The YASFIIRE proxy server injects into Web pages JavaScript dedicated to collecting coordinates of specified elements of a page DOM structure. The collected coordinates are stored in the experiment data base and are later used in eye-tracking data analysis to automatically locate what participants were looking at. These detailed records enable a more comprehensive analysis of users' eye movement patterns in searching behavior.

4.3 Integration of new content with Web pages

Integration of external data into Web pages from a third-party sever can be problematic due to security enforcement inside Web browsers. Our proxy server supports integration of data from third-party sources with Web-based task interfaces from remote servers. Our server can extract the data from any third-party Web page, integrate it and display it together with a Web page from the remote server. This technique avoids cross-domain security issues, that is, it avoids a web browser security checks that forbid a web page from website A request any content from a website B server. This functionality enables experiments that require a high-degree of web page content customization from different data sources.

For example, in one of our IIR experiments, we present a category list on the left side of the SERP using the data which comes from a third Web page. In order to do this, the Request/Response Processor is designed to request and parse a third-party Web page(s) and to cache responses with search results. It then integrates the data extracted from the third-party Web page with the SERP and presents it together to a user (**Figure 5**).

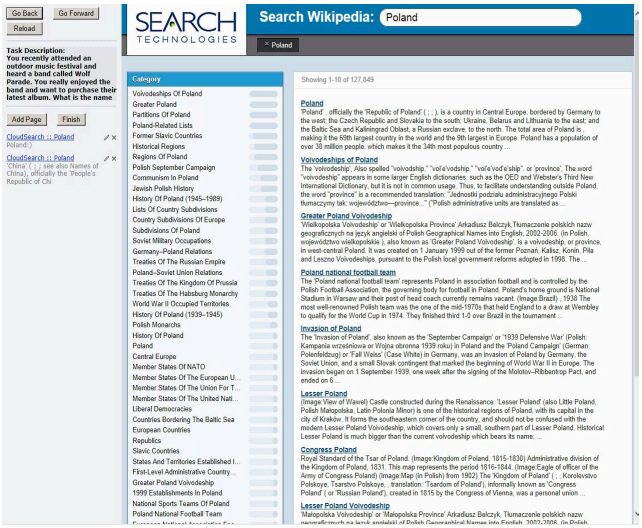

Figure 4. A screen shot showing task description, bookmarking facility and an result of integrating search task interface with additional information (categories).

4.3.1 Example scenarios of use

YASFIIRE architecture and its functionality enables several types of IIR studies. Below we show two example scenarios.

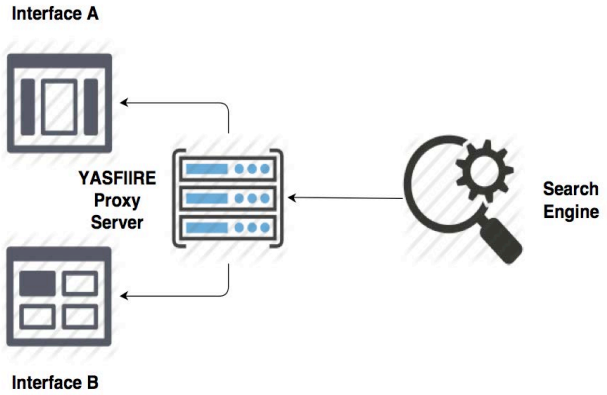

Figure 5. Scenario 1: Search task interface A/B testing with one search engine.

In scenario 1, data (e.g., search results) provided by one external source is displayed to a user in two different task specific interfaces. One application of this scenario is in a standard usability A/B tests of two interfaces.

In scenario 2 (**Figure 6**), data from multiple external sources is displayed using the same task interface. One application of this scenario is in blind testing of search engine results ranking, where identity of a search results source is not revealed to a user. Another application is in experiments in which search results provided by multiple search engines are fused into one set and displayed in one SERP.

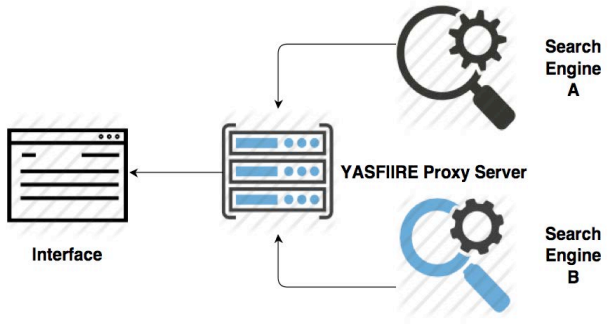

Figure 6. Scenario 2: Search engine algorithms A/B testing with the same search task interface.

5. CONCLUSIONS AND FUTURE WORK

We presented YASFIIRE – Yet Another System for IIR Evaluation. Our system supports IIR research by providing a platform for managing experiments which aim to evaluate and compare user interaction with different interactive information retrieval systems.

The main contributions of YASFIIRE are two-fold. The system architecture enables a number of different experiment scenarios. For example, researchers can not only perform A/B usability testing of different searching engine interfaces, as shown in **Figure 5**, but also compare the algorithms of different information retrieval system (see **Figure 6**).

The second contribution lies in facilitation of eye-tracking data analysis in the realm of IIR. As mentioned in the preceding sections, automated recording of interface element coordinates enables researchers to analyze and interpret data related to participants' eye movement during information search sessions.

In addition to the two main contributions, YASFIRRE has a flexible design based on JavaScript environment. Researchers can tailor the system according to their experimental design needs by "inserting" third party software and by integrating it with specifically designed programs. For instance, in one of our IIR studies, cognitive tasks and interfaces were incorporated into the system. Our PHP Database server integrated cognitive tasks results into our main experiment data database. In this sense, YASFIRRE has a great capacity of assisting researcher in designing a wide range of IIR experiments. YASFIIRE controls experiment tasks using URLs. It can thus be used in combination with any basic usability/eye-tracking logging software that supports Web stimuli.

Limitations of our system include the need for customization specific to each IIR experiment. Also, our system targets (relatively) experienced IIR researchers.

YASFIIRE is a work in progress. In future work, we plan to provide some support for customization and to make our system easier to use for less experienced researchers.

6. ACKNOWLEDGMENTS

This research project was funded, in part, by the IMLS Career Development Grant #RE-04-11-0062-11A awarded to Jacek Gwizdka.

7. REFERENCES

1. Beymer, D. and Russell, D.M. WebGazeAnalyzer: a system for capturing and analyzing web reading behavior using eye gaze. *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, ACM (2005), 1913–1916.
2. Bierig, R., Gwizdka, J., Cole, M., and Belkin, N.J. An Experiment and Analysis System Framework for the Evaluation of Contextual Relationships. *Proceedings of the 2nd International Workshop on Contextual Information Access, Seeking and Retrieval Evaluation*, (2010), 5–8.
3. Buscher, G., White, R.W., Dumais, S., and Huang, J. Large-scale analysis of individual and task differences in search result page examination strategies. *Proceedings of the fifth ACM international conference on Web search and data mining*, ACM (2012), 373–382.
4. Granka, L.A., Joachims, T., and Gay, G. Eye-tracking analysis of user behavior in WWW search. *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM (2004), 478–479.
5. Hall, M.M. and Toms, E. Building a Common Framework for IIR Evaluation. In P. Forner, H. Müller, R. Paredes, P. Rosso and B. Stein, eds., *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*. Springer Berlin Heidelberg, 2013, 17–28.
6. Kelly, D. Methods for Evaluating Interactive Information Retrieval Systems with Users. *Found. Trends Inf. Retr. 3*, 1—2 (2009), 1–224.
7. Lorigo, L., Haridasan, M., Brynjarsdóttir, H., et al. Eye tracking and online search: Lessons learned and challenges ahead. *Journal of the American Society for Information Science and Technology 59*, 7 (2008), 1041–1052.
8. Renaud, G. and Azzopardi, L. SCAMP: A Tool for Conducting Interactive Information Retrieval Experiments. *Proceedings of the 4th Information Interaction in Context Symposium*, ACM (2012), 286–289.
9. Toms, E.G., Freund, L., and Li, C. WiIRE: the Web interactive information retrieval experimentation system prototype. *Information Processing & Management 40*, 4 (2004), 655–675.
10. Tran, V.T. and Fuhr, N. Using eye-tracking with dynamic areas of interest for analyzing interactive information retrieval. *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, ACM (2012), 1165–1166.