

A Pluggable Interactive IR Evaluation Work-bench

ABSTRACT

Information Retrieval (IR) has benefited from standard evaluation practices and re-usable software components, that enable comparability between systems and experiments. However, Interactive IR (IIR) has had only very limited benefit from these developments, in part because experiments are still built using bespoke components and interfaces. In this paper we propose a flexible workbench for constructing IIR experiments that will standardise aspects of the IIR experiment process to improve the comparability and reproducibility of IIR experiments.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces

Keywords

evaluation, framework, standardisation

1. MOTIVATION

Information Retrieval (IR) has benefited from standard evaluation practices and re-usable software components. The Cranfield-style evaluation methodology enabled evaluation programmes such as TREC, INEX, or CLEF. At the same time provision of re-usable software components such as Lucene¹, Terrier², Heritrix³, or Nutch⁴ have enabled IR researchers to focus on the development of those components directly related to their research. However, Interactive IR (IIR) as had only very limited benefit from these developments.

Typically IIR research is still conducted using a single system in a laboratory setting in which a researcher observed

¹<https://lucene.apache.org/>

²<http://terrier.org/>

³<https://webarchive.jira.com/wiki/display/Heritrix/Heritrix>

⁴<http://nutch.apache.org/>

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ACM X-XXXXX-XX-X/XX/XX.

and interacted with a participant [5], usually using a bespoke IIR interface. Developing and running such experiments is a time-consuming, resource exhaustive and labour intensive process [6]. As a result of this bespoke approach, the comparability of IIR experiments and their results suffers. Where studies of the same activities show divergent results, it is difficult to determine whether the differences are due to the specific aspect of IIR under investigation, or simply due to different participant samples or small differences in how the non-investigated user-interface (UI) components were implemented. The bespoke nature also makes it harder to replicate studies, as publications frequently do not contain sufficient detail to exactly replicate the experiment.

In [3] we have proposed a flexible, standardised IIR evaluation framework that aims to address the issues created by variations in the experimental processes and by how context information is acquired from the participants. However, the framework makes no provisions towards providing standardised IIR components that would improve the comparability of the experiment itself, the ease of setting up the experiment, and the ease of reproducibility.

A number of attempts at developing a configurable, re-usable IIR evaluation system have been made in the past. In 2004, Toms, Freund and Li designed and implemented the WiIRE (Web-based Interactive Information Retrieval) system [?], which devised an experimental workflow process that took the participant from information page through a variety of questionnaires and the search interface. Used in TREC 11 Interactive Track, it was built using typical Microsoft Office desktop technologies, which severely limited its capabilities. The system was re-created for the web and successfully used in INEX2007 [7], but lacked flexibility in setup and data extraction. More recently, SCAMP (Search ConfigurAtor for experiMenting with PuppyIR) was developed by Renaud and Azzopardi [4] which is used to assess IR systems, but does not include the range of IIR research designs that are typically done. Another development is CIRSE [?], but to our knowledge it has not been formally implemented or publicly released. These approaches all highlight the difficulty of balancing the two main factors that a such a system must deal with in order to find wide-spread use:

- sufficient flexibility to support the wide range of IIR interfaces and experiments;
- sufficiently simple to implement that it does not increase the resource commitment required to set up the experiment.

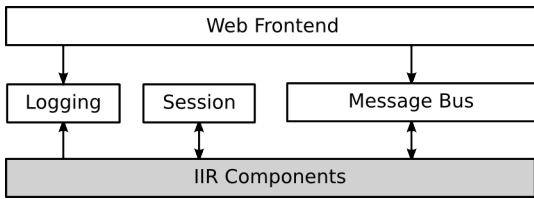


Figure 1: The evaluation workbench consists of the four core modules (Web Frontend, Message Bus, Session, and Logging) into which the IIR components used in the experiment, are plugged.

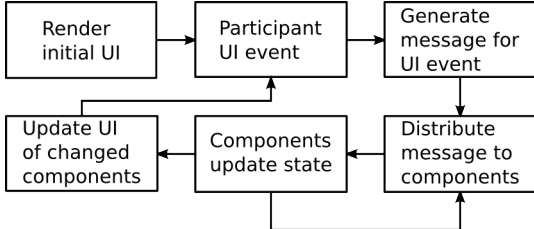


Figure 2: The workbench’s main workflow starts with the generation of the initial UI and then waits for the participant to generate a UI event. The event is processed, the affected component’s state and UI are updated and the workbench goes back to waiting for the next UI event. A powerful aspect of the workflow is that components when they receive a message, can generate their own messages.

2. DESIGN

To achieve the goal of developing a system that fulfils these requirements, we propose a system design that is based around a very lean core into which the researcher can plug the IIR components they wish to include in their experiment. We have implemented this design in our web-based evaluation framework (fig. 1). To achieve maximum flexibility, the system was designed using a message-passing architecture that consists of the following four components:

- **Web Frontend** handles the interface between the participant’s browser and the evaluation workbench and is implemented using a combination of client-side and server-side functionality.
- **Message Bus** handles the inter-component communication and forms the core of the system. It is responsible for passing messages from the **Web Frontend** to the IIR components configured to be listening for those messages and also for passing messages directly between the components.
- **Session** handles loading and saving the components’ current state for a specific participant, hiding the complexities of web-application state from the individual components.
- **Logging** provides a standardised logging interface that allows the components to easily attach logging information to the UI event generated by the participant.

When the researcher sets up the workbench for their experiment, they can freely configure which components to

```
[SearchResults]
handler = application.components.SearchResults
name = search_results
layout = grid-9 vgrid-expand
connect = search_box:query
```

Figure 3: Example configuration for a *Standard Results List* component, showing how the component’s layout (9 grid-cells wide and vertically expanding) and connections to other components (to the “search_box” component via the query message) are specified.

use, how to lay them out, and which components to connect to which other components. Based on this configuration the **Web Frontend** generates the initial user-interface that is shown to the participants. Then, when the participant interacts with a UI element (fig. 2), the resulting UI event is handled by the **Web Frontend**, which generates a message based on the UI event. This message is passed to the **Message Bus**, which uses the configuration provided by the researcher to determine which components to deliver the message to. The components that are listening for that message update their own **Session** state based on the message and then mark themselves as *changed*. After message processing has been completed for all components, the **Web Frontend** then updates the UI for each of the *changed* components.

An example of the configuration used to set-up the experiment is shown in figure 3 (from the experiment in figure 4), specifying the configuration of the “search_results” component. It specifies that the component should be displayed 9 grid-cells wide (the application layout uses a 12-by-12 cell grid layout) and should expand vertically to use as much space as is available. The component is configured to be connected to the “search_box” component via the “query” message. It is this ability to freely plug components together that, we believe, makes the framework sufficiently flexible to support the wide range of IIR experiments, while remaining simple to set-up and use.

3. STANDARD COMPONENTS

The core system provides only the framework into which the IIR components can be plugged. However, to simplify the process of using the work-bench, we provide a set of default components for core IR UI elements. This has the additional advantage that as their behaviour is consistent across experiments, the comparability of experiments using the framework is improved.

3.1 Search Box

The *Search Box* component ([8], p. 49, “Formulate Query Interface” [2], p. 76) provides a standard search box. When the participant enters text and clicks on the “Search” button, it generates a **query** message. It can also be configured to listen to **query** messages from other components, for example if the user can click on document meta-data to search for similar items and the researcher wants to make the participant aware of the query used for the similar items.

3.2 Standard Results List

The *Standard Results List* component ([8], p. 50, “Examine Results Interface” [2], p. 77) provides a default 10 item listing of search results. The *Standard Results List* includes support for displaying snippets ([8], p. 51) and what Wilson calls “Usable Information” ([8], p. 51) for each result document. Unlike the other standard components, which can be used out-of-the-box, the *Standard Results List* has to be extended by the researcher in order to be able to access the search-engine used to power the UI.

3.3 Pagination

To enable the participant to navigate through a set of results, the *Pagination* component ([8] p. 70) displays a configurable number of pages around the current page. When the participant clicks on a page number, the *Pagination* sends a `start` message with the rank of the first document to paginate to. This is usually connected to the *Standard Results List*, which then displays the search results for that page.

3.4 Category Browsing

The *Category Browsing* component ([8], p. 54) provides a hierarchical category structure that the participant can use to explore a collection. Clicking on a category sends a `query` message with the category’s identifier. This is then connected to the *Standard Results List* component, which then shows the documents that have been assigned to that category.

3.5 Saved Documents

The *Saved Documents* component provides an area where the participant can save things that they have found interesting, to support them in their current task. Documents are added through a `save_document` message. The *Saved Documents* component supports an optional tagging feature enabling the participant to tag the document with values specified by the researcher. This can be used to let the participant specify why they have chosen that document or how much it helps them in their current task.

3.6 Task

The *Task* component provides a static display of the task information to show to the user. Two versions of this component are provided, one that displays a static text set in the configuration, and one that can fetch a task description from the database, based on a parameter passed to it.

4. APPLICATION

The evaluation work-bench has so far been used to build two IIR experiments, very different in their nature, clearly demonstrating the work-bench’s flexibility.

The first experiment (fig. 4) re-uses the standard *Task*, *Search Box*, *Pagination*, and *Saved Documents* components, and extends the *Standard Results List* to work with the specific search backend. This set-up re-creates what is essentially a relatively standard search UI configuration, that is being used to investigate query session behaviour.

The second experiment (fig. 5) demonstrates a much richer interface, with more modifications to the components and an experiment-specific component. It re-uses the *Task* and *Category Browsing* components, extends the default *Search Box*, *Pagination*, *Standard Results List*, and *Saved*

Documents components, and adds a new *Item View* component. The message-passing nature of the system made it possible to quickly integrate the new component, so that when the participant clicks on a meta-data facet in the *Item View*, a `query` message is sent to the *Standard Results List* to find items with the same bit of meta-data. The interface was used to investigate un-directed exploration behaviour in a large digital cultural heritage collection.

5. WHERE TO GO NEXT?

The stated aim of this paper was to present a novel, plugable, extensible, and configurable IIR evaluation work-bench, that supports our wider aim of improving IIR experiment comparability. We have presented our web-based IIR evaluation workbench, which is sufficiently flexible to support the wide range of web-based IIR experiments that are undertaken, while being sufficiently simple and light-weight to encourage wide-spread use of the workbench.

To enable this wide-spread use, we will be freely releasing the system in the near future. We are also moving to engage with the wider research community to determine to what degree the work-bench satisfies their needs for an evaluation system and what needs to be done to achieve the wide-spread use needed to improve IIR experiment comparability.

6. REFERENCES

- [1] R. Bierig, M. Cole, J. Gwizdka, N. J. Belkin, J. Liu, C. Liu, J. Zhang, and X. Zhang. An experiment and analysis system framework for the evaluation of contextual relationships. In *CIRSE 2010*, page 5, 2010.
- [2] C. Chua. A user interface guide for web search systems. In *Proceedings of the 24th Australian Computer-Human Interaction Conference, OzCHI '12*, pages 76–84, New York, NY, USA, 2012. ACM.
- [3] M. M. Hall and E. G. Toms. Building a common framework for iir evaluation. In *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 4th International Conference of the CLEF Initiative - CLEF 2013*, 2013.
- [4] G. Renaud and L. Azzopardi. Scamp: a tool for conducting interactive information retrieval experiments. In *Proceedings of the 4th Information Interaction in Context Symposium*, pages 286–289. ACM, 2012.
- [5] J. Tague-Sutcliffe. The pragmatics of information retrieval experimentation, revisited. *Information Processing & Management*, 28(4):467–490, 1992.
- [6] E. G. Toms, L. Freund, and C. Li. Wiire: the web interactive information retrieval experimentation system prototype. *Information Processing & Management*, 40(4):655–675, 2004.
- [7] E. G. Toms, H. O’Brien, T. Mackenzie, C. Jordan, L. Freund, S. Toze, E. Dawe, and A. Macnutt. Task effects on interactive search: The query factor. In *Focused access to XML documents*, pages 359–372. Springer, 2008.
- [8] M. L. Wilson. *Search User Interface Design*, volume 20. Morgan & Claypool Publishers, 2011.

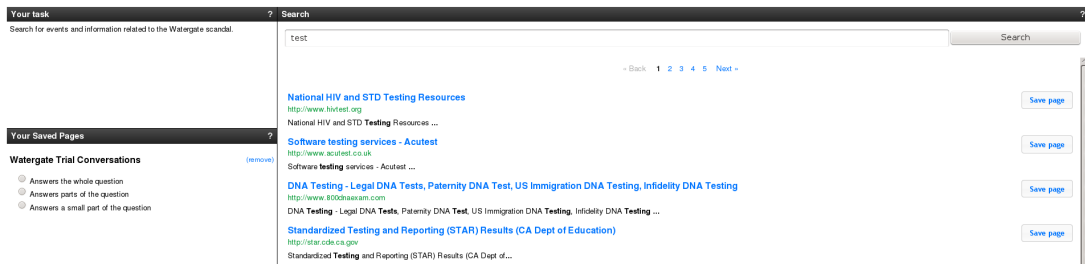


Figure 4: Screenshot showing an experiment with a very basic configuration consisting of *Task*, *Search Box*, *Pagination*, *Standard Results List*, and *Saved Documents* components. This is being used to investigate query behaviour for tasks that require query reformulations.

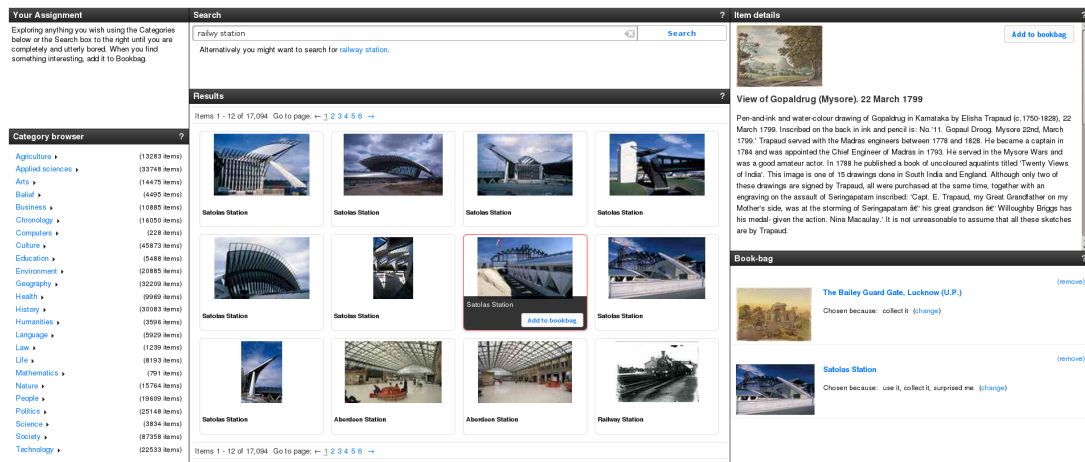


Figure 5: Screenshot showing an experiment that makes heavy use of the customisation options offered by the workbench. This configuration was used to investigate un-directed exploration in a digital cultural heritage collection.