TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments

Tim Gollub, Benno Stein, Steven Burrows, Dennis Hoppe
Bauhaus-Universität Weimar
99421 Weimar, Germany
<first name>.<last name>@uni-weimar.de

Abstract—**With its close ties to the Web, the information retrieval community is destined to leverage the dissemination and collaboration capabilities that the Web provides today. Especially with the advent of the software as a service principle, an information retrieval community is conceivable that publishes executable experiments by anyone over the Web. A review of recent SIGIR papers shows that we are far away from this vision of collaboration. The benefits of publishing information retrieval experiments as a service are striking for the community as a whole, including potential to boost research profiles and reputation. However, the additional work must be kept to a minimum and sensitive data must be kept private for this paradigm to become an accepted practice. In order to foster experiments as a service in information retrieval, we present the TIRA (Testbed for Information Retrieval Algorithms) web framework that addresses the outlined challenges and possesses a unique set of compelling features in comparison to existing web-based solutions. To describe TIRA in a practical setting, we explain how it is currently used as an official evaluation platform for the well-established PAN international plagiarism detection competition. We also describe how it can be used in future scenarios for search result clustering of non-static collections of web query results, as well as within a simulation data mining setting to support interactive structural design in civil engineering.**

I. Motivation and Survey

In all scientific disciplines, researchers strive for a detailed specification of how they designed their experiments in order to achieve their published results. For modern computer science disciplines, where experiments are made on top of an advanced stack of hardware and software components, providing all the necessary information within the scope of a paper is commonly impossible. In this regard, reproducing results from published experiment descriptions in order to improve upon published baselines is cumbersome.

Within information retrieval research, the integration of previous work in own experiments is simplified if the data and software assets are published with the papers. In this respect, Armstrong et al. [1] pointed out the verification problems that arise if research assets are not streamlined with the latest publications. In our view, the most convenient way to enhance comparability in experiments is to publish experiments as an online service where researchers can verify experiment results and explore alternate parameter settings. In order to explore how information retrieval research is published today, we reviewed all 108 full papers from the 30 sessions of the SIGIR 2011 proceedings concerning the experiment assets that were used as part of the research. The results in Figure 1 show the extent to which the authors published their *data*, their *software*, and whether the experiments were hosted as a *service*. All three aspects need to be addressed to make publishing data and software more widespread. The three questions we posed when examining the proceedings and our findings are given as follows.

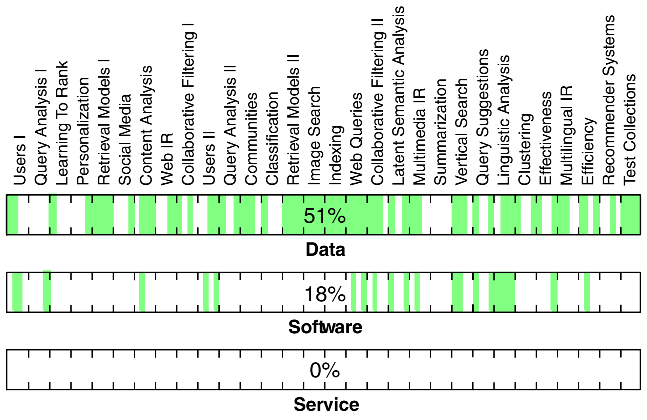


Figure 1: Assessment of the 108 full papers at SIGIR 2011 with respect to the publication of data, software, and experiments hosted as a service. The papers are ordered as they appear in the conference proceedings, and the bins on the horizontal axis denote the 30 sessions. The colored bars denote papers that meet the respective criteria.

Data. *Are any of the datasets used in the research publicly available?* We observe that 51% of the papers use a publicly available dataset for experiments. In this respect, evaluation initiatives such as TREC do a wonderful job in supplying researchers with shared datasets. All papers from the Image Search, Indexing, Retrieval Models, Test Collections, and Web Queries sessions used public datasets. Other datasets such as search engine query logs require *non-disclosure* conditions due to commercial and sensitivity reasons, which accounts for much of the unavailability of data. Correspondingly, only two of the eight Query Analysis papers used public datasets, which was second lowest and only ahead of the Summarization papers, where no public datasets were used.

Software. *Are any of the software assets used in the research publicly available?* We observe that 18% of the papers use shared software contributions, and hence publishing software is currently a minority practice in information retrieval research. Only the Linguistic Analysis papers published accompanying software in all cases. One explanation for this low ratio is the *lack of acknowledgment* researchers get for publishing software. Some recent conferences are reviewing software contributions as a part of paper review processes, but new incentives must be created that turn invested time in developing reusable software into reputation gain for the researchers to foster further software release.

Service. *Are any of the experiments in the research provided as an online service?* We observe that providing experiment software as a service is not practiced in the examined papers. The application closest to a web service is the ViewSer [2] demonstration. The lack of a *compelling web framework* that researchers can use to transform their experiments into a web service is a plausible explanation.

II. Related Work and Design Goals

Motivated by the observations from the paper study above, we propose the development of the TIRA online framework to foster publishable information retrieval experiments. The TIRA project is based on the needs for (1) local instantiation, (2) web dissemination, (3) platform independence, (4) result retrieval, and (5) peer-to-peer collaboration. Our assessment of existing web-based experimentation frameworks in Table I with respect to these goals shows that no system fully complies.

Table I: Assessment of existing web-based experimentation frameworks with respect to our five proposed design goals.

Tool	Domain	1	2	3	4	5
evaluatIR[1]	IR	×	✓	✓	✓	×
expDB[2]	ML	×	×	×	✓	×
MLComp[3]	ML	×	✓	×	✓	×
myExperiment[4]	any	×	✓	✓	✓	×
NEMA[5]	IR	×	✓	×	✓	×
TunedIT[6]	DM, ML	✓	✓	×	✓	×
Yahoo! Pipes[7]	Web	×	✓	×	×	×

[1]http://www.evaluatir.org/
[2]http://expdb.cs.kuleuven.be/
[3]http://www.mlcomp.org/
[4]http://www.myexperiment.org/
[5]http://www.music-ir.org/
[6]http://www.tunedit.org/
[7]http://pipes.yahoo.com/

Local Instantiation. If data must be kept confidential, the framework must be able to reside with the data, hence the framework must be locally installable. Unlike centralized experiment platforms like MLComp and myExperiment, local instantiation allows experiments on sensitive data to be published as a service from a local host. External researchers can then use the service for comparison and evaluation of their own research hypotheses, whilst the experiment provider is in full control of the experiment resources.

Web Dissemination. URLs are definitive identifiers for digital resources. With a URL, researchers can conveniently link the results in a paper with the experiment service used to produce them. Especially for standard pre-processing tasks or evaluations on private data, a web service can become a frequently cited resource. In addition, attention can be attracted to one's work through integration of the service into home pages and blog articles. To address the issue of digital preservation, URLs should encode all information needed to recompute a resource, such as program and input parameter specifications, in case stored data is lost.

Platform Independence. The sophisticated and varying software and hardware requirements of information retrieval experiments as well as individual coding preferences of software developers render any development constraints imposed by the web framework critical for its success. Ideally, software developers can deploy experiments as a service unconstrained by the utilized operating system, parallelization paradigm, programming language, or data formats. Local instantiation is one key to realize this goal. Furthermore, the web framework must operate as a layer strictly on top of the experiment software and should use, instead of close intra-process communication such as in TunedIT, standard inter-process communication on the POSIX level and the file system to exchange information. This way, any running software can be deployed as a web service without internal modifications.

Result Retrieval. Especially for retrieval tasks with high computational costs, the maintenance of a public result repository can become a valuable asset of a research group. For example, experiment services that can index datasets with state-of-the-art natural language processing technology have the potential to raise the comparability of retrieval model research to a higher level. For clustering and result diversification research, comparability is enhanced by establishing static snapshots of the search results from major search engines regularly. The persistent storage of experiment results by the web framework is key to achieve this goal. Even if the public release of an experiment service is not desired, the framework is still useful if it assumes responsibility for managing the raw experiment results and making them available across a research team.

Peer-to-Peer Collaboration. Consider a scenario where a consortium of service providers become renowned *gatekeepers* for various streams of research, and maintain the community-wide repository of state-of-the-art algorithms, datasets, and experiment results on their web site. The gatekeepers drive the standardization of data formats and can, by utilizing the retrieval facility, stage competitions in a semi-automated fashion. A mechanism for connecting the local framework instances to a network of experimentation nodes has to be provided to achieve this scenario. It should be noted that none of the existing web-based experimentation platforms implement peer-to-peer collaboration.

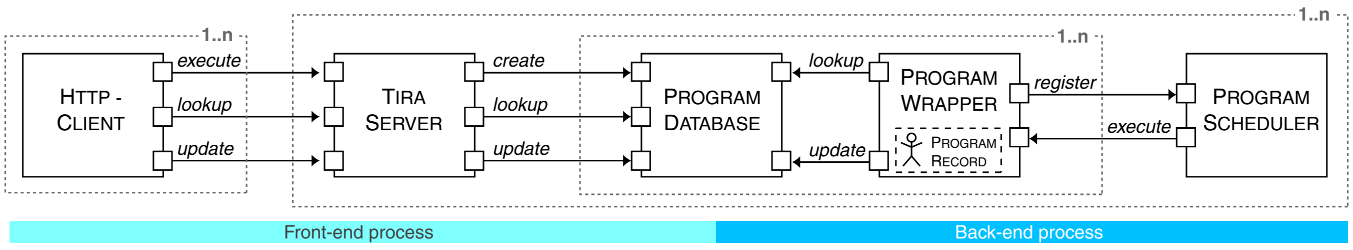


Figure 2: Component diagram of TIRA. Towards the left, the front-end process dealing with the user-interaction is illustrated. To the right, the back-end program execution process is shown. Requests are illustrated by arrows and imply a response from the requested component.

III. System Architecture

The basic functionality of TIRA is to take a locally executable program and turn it into a web service. To use TIRA for this purpose, the software is first downloaded and instantiated on the local computing infrastructure. System compatibility should not become an issue here, since we distribute TIRA as an executable Java JAR file.[1] For the deployment of new programs, TIRA requires a program specification file in JSON format: the *ProgramRecord*, as shown in Figure 2. In its minimal form, the *ProgramRecord* comprises (1) a unique name for the program, (2) the generic structure of the program execution command, (3) the value range of each input parameter that affects the output of the program, and (4) the names of all important output files. An example of a generic program execution command and its respective input parameter specification is given in Figure 3. In general, more complex commands are possible that concatenate multiple programs via UNIX-pipes or define parameter substitutions that produce non-terminals (further parameters).

Provided with the information in the *ProgramRecord*, TIRA instantiates and updates all system components that are needed to establish a web service for the new program. All system components are shown in Figure 2. The operating principle of TIRA can be described as two major processes: the front-end process dealing with user interaction, and the back-end process dealing with program execution. As indicated in the component diagram, the *ProgramDatabase* takes on a special role in TIRA's system architecture, since it links the two processes together. The *ProgramDatabase* is instantiated for each *ProgramRecord* individually, it stores past and pending program runs, and it indexes the input parameters of the runs to provide basic retrieval functionality. Note that besides the default local database, TIRA can also connect to a database on a foreign TIRA instance to accomplish peer-to-peer collaboration. The front-end and back-end processes are unaffected by this distinction. In the remainder of this section, the components of these two processes are described from the back-end process first, followed by the front-end process lastly.

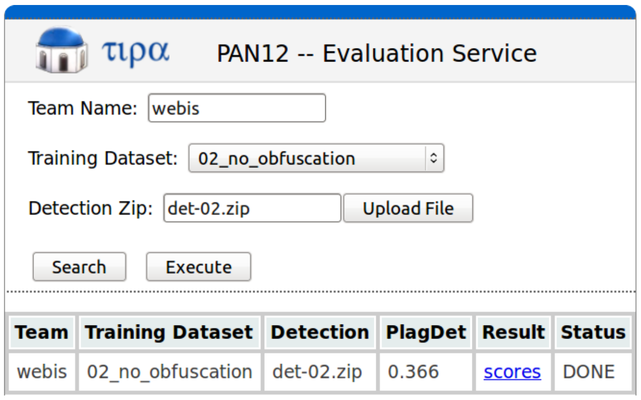
[1]See http://tira.webis.de for the latest TIRA release information.

```
python myexp.py $param1 $param2 > result.txt
$param1 -> a | b | c
$param2 -> [0-9]+
```

Figure 3: BNF grammar for a Python program "myexp" with two input parameters for execution in TIRA.

The back-end process involves the *ProgramWrapper* and *ProgramScheduler* components. For each *ProgramRecord*, an individual *ProgramWrapper* is instantiated to continuously query its associated *ProgramDatabase* for pending program runs. Given that TIRA instances might be equipped with different resources in a collaborative environment, the lookup request sent may contain constraints with respect to accepted input parameter values. When a matching program run is received, the *ProgramWrapper* registers this at the *ProgramScheduler* for addition to an execution queue. The *ProgramScheduler* keeps a pool of system threads, which continuously take the next run in the queue and request its execution. To start the program, the generic command in the *ProgramRecord* is substituted with the run-specific values and is called inside a run-specific working directory. During execution, the *ProgramWrapper* listens on the error output stream and updates the database with notifications and results, which then become available to the front-end process.

The *HttpClient* and *TiraServer* components are associated with the front-end process. The *HttpClient* is usually a web browser controlled by a TIRA user, but also a TIRA instance may fill this role to communicate with other TIRA instances. For each *ProgramRecord*, the *HttpClient* can access a web page on the *TiraServer* via a program-specific URL (e.g. `http://<domain>/program/myexp`). A screenshot of a TIRA web page is given in Figure 4. The TIRA web page features the program input parameters as HTML form elements, and offers functionality for retrieving program runs with specific parameter values (Search) and for executing new runs (Execute). The result table at the bottom contains the current execution status, and the results of all executed program runs are displayed. If the value range of an input parameter is specified in the *ProgramRecord* as an enumeration (cf. `$param1` in Figure 3), the input values are listed in a selection box. Otherwise in the case of an intrinsic definition (cf. `$param2` in Figure 3), a text input field is given instead.



Team Name:	webis
Training Dataset:	02_no_obfuscation
Detection Zip:	det-02.zip

Team	Training Dataset	Detection	PlagDet	Result	Status
webis	02_no_obfuscation	det-02.zip	0.366	scores	DONE

Figure 4: Screenshot of a TIRA web page for the PAN competition 2012. On the web page, PAN participants specify a dataset and upload their plagiarism detection results. On execute, TIRA runs an evaluation script and displays the performance assessment for the submission.

As a third option, TIRA allows submission files as input parameters, in which case a file upload element is shown to the user.

To retrieve specific program runs, the TIRA user can specify a subset of the input parameters and submit the HTML form by clicking the Search button. The *TiraServer* looks up the database and returns a web page with the matching results. For retrieval requests, the form is submitted using the HTTP GET method, which means that all form values are encoded into the URL. This URL can thus be used for the dissemination of results as discussed in Section II. In case all input parameters are populated with valid values, the execution of the program can also be requested. Note that the *TiraServer* handles multiple values for parameters by generating an independent program run for each possible combination of values. This gives TIRA users a convenient means to execute a series of runs with a single parameter specification. In case a combination of parameter values has not been seen before, a new program run is created with a pending status and stored in the database. Responsibility for the pending run is handed to and executed by the back-end process.

IV. System Scenarios

This section outlines three of our ongoing group projects where TIRA is involved. The plagiarism detection, search result clustering, and simulation data mining scenarios are chosen to exemplify the applicability of TIRA in a wide variety of contexts.

A. *Plagiarism Detection*

Automated plagiarism detection involves the heuristic retrieval of candidate plagiarism matches, and the detailed comparison of each pair. The evaluation of plagiarism detection algorithms has been featuring prominently in the PAN series of international plagiarism detection competitions [3, 4, 5]. At PAN, participants are asked to develop plagiarism detection algorithms on large collections of training data, for evaluation on similarly large test collections. The competition organizers then evaluate the submissions using a combined measure of precision, recall, and granularity.

For the PAN plagiarism detection competition in 2012, TIRA is employed to create up-to-date alternatives for the organization of the competition.[2] TIRA is used for processing results during the competition training phase, where participants can submit their results. TIRA then evaluates the result submissions, provides instant feedback, and stores the submissions for later retrieval.

In order to take part in the final performance assessment, the participants use TIRA to submit their detection algorithms as executable software on either a Windows or Linux-based virtual machine. All submitted software is directly deployed to TIRA to perform a distributed evaluation on the holdout test set. This course of action is desired, since the test set contains data that is subject to non-disclosure. In addition, TIRA allows the runtime of the submitted approaches to be recorded for the first time.

After the competition, the participants have the opportunity to opt-in for a public release of their plagiarism detection software as a service. In consideration of all of the above benefits, the TIRA web service provides an excellent framework as a competition evaluation forum to make the competition activities highly visible and reproducible.

B. *Search Result Clustering*

Search result clustering addresses the following task: given an informational query, group the ranked list of search results retrieved from a search engine into coherent clusters to reduce the time and effort for finding relevant information. Each cluster is ideally associated with a meaningful cluster label and exclusively represents a particular query aspect.

Recently a proposal was made to eliminate undesired topic repetition in result clusterings by filtering out search results that address query aspects already covered in the top search results [6]. It was also proposed to avoid extreme clusterings in order to facilitate effective navigation: neither the number of cluster labels nor the number of search results per cluster should exceed the size of the result list head [7]. In both papers, a key component for the experiments is the quantification of the query aspect coverage in search results. Since existing strategies for this purpose are evaluated on different non-public datasets, and because search results retrieved from a search engine for a query are dynamic, it is impossible to directly assess the best strategy.

TIRA provides a convenient means to publish a retrieval service for research purposes, where search results can be fetched for a query from multiple search engines and stored as static data resources. On the basis of this TIRA service, search result clustering experiments become out-of-the-box reproducible and accessible by others. Web search engines

[2]http://pan.webis.de

such as Yahoo! and Bing, knowledge resources such as Wikipedia, and news portals such as Yahoo! News and the New York Times can all be consulted. What can be anticipated from this approach is that not only external researchers benefit, but also the service publishers themselves become acknowledged in publications. The fact that each data resource stored by TIRA can be referenced by a unique URL further supports this win-win scenario.

C. Simulation Data Mining

Simulation data mining combines the systems simulation and data mining fields to develop knowledge and intelligence from simulated data, where real data is difficult to obtain or cost-inefficient to generate [8]. A recent project to support interactive bridge design in civil engineering was undertaken where simulation data mining was applied [9]. The project group has been simulating bridge models based on varying geometries and materials to demonstrate an interactive design process for conceptual design phases.

In the cited project, the input parameter space includes material and geometry features from bridge models, the range of values for each parameter, and the number of increments for each parameter. The output parameter space includes displacement, stress, and strain simulation measurements at various coordinates within the bridge models. Given that the multiplicity of the parameter space is enormous, the design of TIRA experiments to facilitate the exploration of subsets of the parameter space is a valuable addition to the project. Therefore the ability to embed parameters in TIRA experiment workflows is key to managing large parameter spaces and big data experiments.

This simulation data mining scenario also highlights how researchers working on TIRA can collaborate more effectively. The project as described is multidisciplinary, and has brought civil engineers and computer scientists together to work on common problems. Given that each discipline has well-established methods and habits for going about their work, the online availability of TIRA facilitates a tighter interaction of these groups to explore new experiment designs and outcomes together. The simple web-based experiment framework of TIRA is key to engage fruitful collaboration between multiple research groups.

V. Summary

In this paper, we proposed the TIRA web framework for information retrieval experiments as a service motivated by low trends in sharing data and software in recent information retrieval research. The fundamental design decisions concerning local instantiation, web dissemination, platform independence, result retrieval, and peer to peer collaboration address the specific needs of information retrieval research. In comparison with existing tools and services of this purpose, TIRA possesses a unique set of features. The current version of TIRA has already been put to widespread use as part of the PAN competition series in 2012. Further application will allow non-static collections of web query results to be processed for search result clustering, and easier collaboration opportunities for researchers of multidisciplinary research projects. The TIRA service and supporting information is available online.[3]

Acknowledgements

This paper is an extended version of the two-page poster “Ousting Ivory Tower Research: Towards a Web Framework for Providing Experiments as a Service” [10].

References

[1] T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel, “Improvements that don’t add up: Ad-hoc Retrieval Results since 1998,” in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM ’09, Hong Kong, China: ACM, 2009, pp. 601–610.

[2] D. Lagun and E. Agichtein, “ViewSer: Enabling Large-Scale Remote User Studies of Web Search Examination and Interaction,” in *Proceedings of the 34th International ACM Conference on Research and Development in Information Retrieval*, SIGIR ’11, Beijing, China: ACM, 2011, pp. 365–374.

[3] M. Potthast, B. Stein, A. Eiselt, A. Barrón-Cedeño, and P. Rosso, “Overview of the 1st International Competition on Plagiarism Detection,” in *SEPLN 09 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, 2009, pp. 1–9.

[4] M. Potthast, A. Barrón-Cedeño, A. Eiselt, B. Stein, and P. Rosso, “Overview of the 2nd International Competition on Plagiarism Detection,” in *Notebook Papers of CLEF 10 Labs and Workshops*, 2010.

[5] M. Potthast, A. Eiselt, A. Barrón-Cedeño, B. Stein, and P. Rosso, “Overview of the 3rd International Competition on Plagiarism Detection,” in *Notebook Papers of CLEF 11 Labs and Workshops*, 2011.

[6] B. Stein, T. Gollub, and D. Hoppe, “Beyond Precision@10: Clustering the Long Tail of Web Search Results,” in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 11)*, ACM, 2011, pp. 2141–2144.

[7] T. Gollub, B. Stein, and D. Hoppe, “Search Result Presentation Based on Faceted Clustering,” in review, 2012.

[8] H. Cunningham, N. Fuhr, and B. Stein, “Challenges in Document Mining (Dagstuhl Seminar 11171),” *Dagstuhl Reports*, vol. 1, no. 4, pp. 65–99, 2011.

[9] S. Burrows, B. Stein, J. Frochte, D. Wiesner, and K. Müller, “Simulation Data Mining for Supporting Bridge Design,” in *Proceedings of the 9th Australasian Data Mining Conference*, CRPIT, vol. 121. Ballarat, Australia: ACM, 2011, pp. 163–170.

[10] T. Gollub, B. Stein, and S. Burrows, “Ousting Ivory Tower Research: Towards a Web Framework for Providing Experiments as a Service,” in *Proceedings of the 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12)*, B. Hersh, J. Callan, Y. Maarek, and M. Sanderson, Eds., Aug. 2012.

[3]http://tira.webis.de