First Experiences with TIRA for Reproducible Evaluation in Information Retrieval

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

ABSTRACT

The verifiability and comparability of computational experiments is a major shortcoming in scientific publications, even at top conferences. In recent years, various services emerged that try to address this problem by providing a global platform where researchers can upload programs along with experiment results. However, these platforms are not well accepted, partly due to their inherent top-down character: a single institution prescribes the formats and technologies to be used. We argue that a community-wide evaluation platform can evolve only from an ongoing bottom-up effort.

For the field of information retrieval we have been undertaking concrete steps to launch and foster this idea with TIRA [4]. Here, we present the concept and an implementation of a web-based experimentation environment that greatly simplifies maintenance and publishing of executable experiments for a research group. TIRA's system architecture retains researcher's full control over their research assets; moreover, no constraints with respect to data formats or programming technologies are prescribed. We see several reasons for researchers to publish their experiments as a web service with TIRA, namely, to simplify their experiment design and execution, to gain credibility, and to easily disseminate results.

This paper reports on experiences from developing TIRA towards our goal. Design goals are reviewed, existing evaluation platforms are analyzed, and the architecture of our current implementation is presented. In particular, we present insights from the first widespread use of TIRA at the PAN series of international plagiarism detection competitions in 2012. Altogether, our review is promising: the design decisions underlying TIRA are both powerful and flexible enough to cope with the widely varying programming preferences of the researchers.

Categories and Subject Descriptors: H.5.3 [Information Systems]: Information Interfaces and Presentation—Group and Organization Interfaces

Keywords: Open Evaluation, Experiment Management, Result Dissemination

1. MOTIVATION

John Ioannidis attracted considerable attention in 2005 with his essay "Why Most Published Research Findings Are False" [5]. Ioannidis argues that research findings published in papers are likely to be biased towards the approaches of the authors, commonly because of selective result reporting and unequal parameter tuning efforts. To improve upon this situation, he concludes that official evaluation initiatives are needed where researchers register their approaches for an objective assessment. In addition, the SWIRL 2012 meeting of 45 information retrieval researchers considered evaluation as a "perennial issue in information retrieval", and that a "community evaluation service" is of specific interest [1]. With initiatives such as TREC[1], CLEF[2], and PAN[3], the information retrieval research community has established evaluation campaigns with great success, with datasets of past campaigns being frequently used for current research.

We see two major limitations of these initiatives that we want to overcome with our open source evaluation platform TIRA (the "Testbed for Information Retrieval Algorithms"). First, it is obvious that the scale of these initiatives cannot address all interesting research questions that arise. To cover the bulk of remaining research questions, a community wide evaluation campaign is needed that is supported by convenient open software. Any researcher must be empowered to easily set up and conduct an evaluation initiative for a specific task of interest. Second, the annual schedule of renowned evaluation initiatives is problematic. In this respect, Armstrong et al. [2] analyzed the performance results achieved outside the official TREC initiative on various TREC collections as published in SIGIR and CIKM papers from 1998 to 2008. The findings showed that the vast majority of these papers are not streamlined with the official TREC results, which in turn leads to a series of false conclusions in the papers and "improvements that don't add up". To avoid the ignorance of existing results, ongoing evaluation initiatives are needed that continuously integrate new results submitted over the Web.

With TIRA, we are developing an open source evaluation platform where we aim to overcome the limitations stated above [3, 4]. The decisive feature of TIRA is that the software can be downloaded by any research group to organize and conduct an evaluation initiative on their local computing infrastructure. For every experiment, TIRA provides a web service through which participants can submit their algorithms or results at any time. TIRA evaluates new submissions automatically by executing the experiment evaluation software provided by the evaluation organizers from the command line of the underlying operating system. All experiment results are stored and indexed in a database, which is queried by the web service to display the current results.

In the remaining sections of the paper, the design goals for TIRA are presented and compared to existing experiment platforms in Section 2, whereas in Section 3 we explain the system architecture of TIRA in detail. In Section 4, we give an experience report of our first significant deployment of TIRA at the PAN plagiarism detection competition, and we provide lessons learned and future recommendations. We then summarize our work in Section 5.

SIGIR 2012 Workshop on Open Source Information Retrieval.
August 16, 2012, Portland, Oregon, USA.

[1] `http://trec.nist.gov`
[2] `http://www.clef-initiative.eu`
[3] `http://pan.webis.de`

2. DESIGN GOALS AND RELATED WORK

Our efforts to make the deployment of TIRA as simple and convenient as possible led to a set of five design goals that we consider as crucial for its widespread use. The design goals are based on the needs for local instantiation, web dissemination, platform independence, result retrieval, and peer to peer collaboration. Our assessment of existing experimentation frameworks with respect to these goals is depicted in Table 1, which shows that none of these systems fully comply.

Table 1: Assessment of existing experimentation frameworks with respect to our five proposed design goals.

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

[1]http://www.evaluatir.org/
[2]http://expdb.cs.kuleuven.be/expdb/
[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/

1. *Local Instantiation.* In case data must be kept confidential, the platform must be able to reside with the data, hence the platform 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.

2. *Web Dissemination.* URLs are definitive identifiers for digital resources. If all runs of an experiment are accessible over a unique 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, such 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.

3. *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 experimentation 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 experimentation 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.

4. *Result Retrieval.* Especially for computationally expensive retrieval tasks, 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 experimentation 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.

5. *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. Note that currently none of the experimentation platforms implements peer to peer collaboration.

3. 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.[4] For the deployment of new programs, TIRA requires a program specification file in JSON format: the *ProgramRecord*, as shown in Figure 1. In its minimal form, the *ProgramRecord* comprises (1) a unique name for the program, (2) the generic structure of the program execution command, and (3) the value range of each input parameter that affects the output of the program. An example of a generic program execution command and its respective input parameter specification is given in Figure 2. 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 1. 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 beginning with the back-end process first, followed by the front-end process lastly.

The TIRA back-end process involves the *ProgramWrapper* and *ProgramScheduler* system components. For each *ProgramRecord*, an individual *ProgramWrapper* is instantiated to query its asso-

[4]See `http://tira.webis.de` for latest TIRA release information.

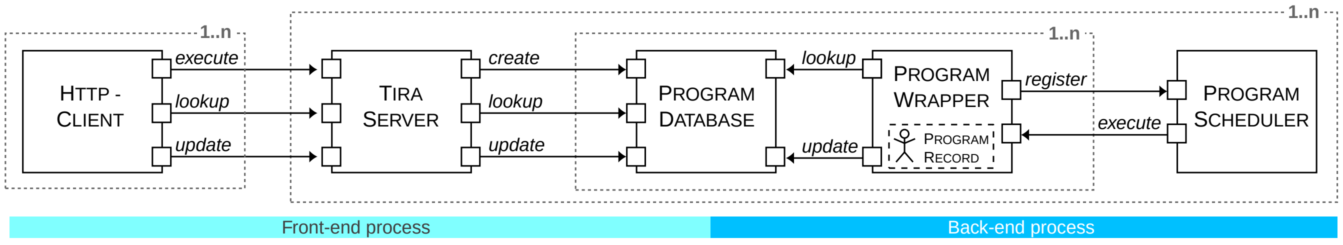


Figure 1: Component diagram of TIRA. Towards the left, the front-end process dealing with the user-interaction is illustrated. To the right, the back-end program execution process is shown. Requests are illustrated by arrows and imply a response from the requested component.

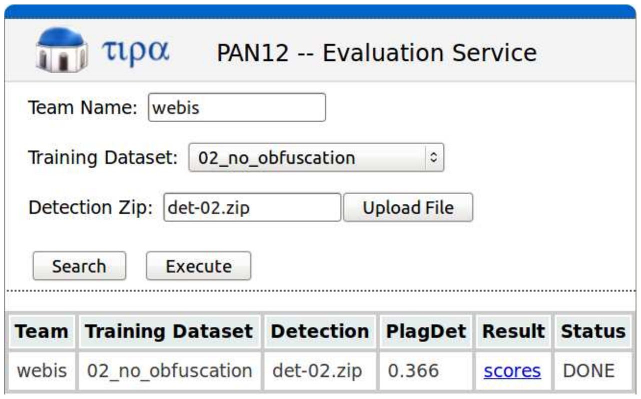
```
python myexp.py $param1 $param2 > result.txt
$param1 -> a | b | c
$param2 -> [0-9]+
```

Figure 2: BNF grammar for a Python program "myexp" with two input parameters for execution in TIRA.

ciated *ProgramDatabase* continuously 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 TIRA front-end process involves the remaining *TiraServer* and *HttpClient* system components. 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 3. 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. `$param1` in Figure 2), the input values are listed in a selection box. Otherwise in the case of an intrinsic definition (cf. `$param2` in Figure 2), a text input field is given instead. 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 2. 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.



PAN12 -- Evaluation Service

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 3: 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.

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.

4. ANALYSIS OF TIRA AT PAN

In this section we report on the first deployment of TIRA in an official evaluation campaign. TIRA has been used as the training and evaluation platform for the "detailed comparison" task of the 2012 international PAN plagiarism detection competition.[5] The competition started with the release of training data in March 2012, and officially ended after the evaluation of the participant submissions in July 2012. For TIRA, its successful deployment in the challenge constitutes an important milestone and was an excellent opportunity to analyze the software under realistic conditions.

The participants of the PAN "detailed comparison" challenge were asked to develop software capable of solving the following task: Given a suspicious document and a potential source document pair, extract and record all plagiarized passages from the suspicious document and the corresponding source passages from the source document. Unlike the previous PAN competitions, the participants of 2012 did not submit their detection results on an unlabeled test set, but instead submitted their software. This strategy allowed a set of real plagiarism cases subject to non-disclosure to be incorporated into the test set to improve the authenticity of the evaluation. In addition, the organizers could evaluate the runtime characteristics of the submitted approaches for the first time.

[5]`http://pan.webis.de/`

Two TIRA services were deployed to support the running of the competition: (1) A service to compute performance scores on the training data, and (2) A service for the evaluation of the software submissions on the private test set. We now describe how TIRA has been used in each of these settings in the remainder of this section.

4.1 Training Phase Evaluation Service

For the training phase, the organizers released a dataset with ground truth to be used by the participants to train their approaches. A TIRA service was provided to evaluate the performance of an approach using the training set. On the TIRA service web page, participants were able to upload their compressed detection results and receive the "PlagDet" performance score in return (cf. Figure 3), which combines aspects of precision, recall, and granularity. To compute PlagDet, the compressed submissions were extracted and evaluated with a Python implementation of PlagDet. The generic execution command used by TIRA for the evaluation was hence: "`unzip -qq -o $det -d det && python perfmeasures.py -p $truth -d det > scores.txt`". The parameters starting with a $-symbol were substituted according to the data provided in the web page input fields similar to the example described in Section 3 and Figure 2.

For the organizers of PAN, the evaluation service provided some feedback about the progress of the participants. In the past competitions, the organizers observed that the majority of participants started working seriously only in the few days before the submission deadline. With the public evaluation service, we hoped to create an atmosphere where participants were motivated by the recorded PlagDet scores to date acting as a leader board. One week prior to the submission deadline, the evaluation service received 12 submissions from two of the eleven final participants. Three further participants started making submissions in the final week, resulting in 38 computed PlagDet scores altogether. The remaining six participants did not use the training phase evaluation service, and may have simply elected to evaluate their training results offline. Although the TIRA service was a useful tool for the participants, we learned that further incentives for its usage must be provided to effectively foster the early tinkering within the competition.

4.2 Test Phase Evaluation Service

In the test phase, TIRA was used to organize and conduct the evaluation of the submitted programs. In total, we received eleven plagiarism detection programs for evaluation on the hidden test set. Coincidentally, eleven "external detection" result sets were submitted in 2011 [6], suggesting that the submission of software was an acceptable demand of the participants. The software received varied greatly with respect to its size, runtime performance, and programming language used, and we received submissions for both Windows and Linux operating systems. In this respect, the system independence of TIRA has been successfully demonstrated. We managed to get all submitted software running, and with the exception of one submission, the output files produced were valid. For each of the submissions, we created a *ProgramRecord* based on the installation manual provided by the participants. Although the programs sometimes demanded inconvenient input specifications for processing the test data, the powerful parameter substitution mechanism of TIRA made the task achievable. To evaluate each submission against the test set, we implemented an additional TIRA service that sends an execution request for every document pair in the test set to the TIRA service of the submission. Here, the web dissemination capability of TIRA is highly convenient.

In the near future we plan to give the PAN 2012 participants the opportunity to opt-in for a public release of their plagiarism detection software as a TIRA service on our computing infrastructure. For future evaluation initiatives, we aim to develop an automated program deployment mechanism for TIRA: Participants download the evaluation resources for a competition and deploy them on a local TIRA instance. Once developing and testing on the local TIRA instance is done, TIRA sends the final *ProgramRecord* and software to the official TIRA evaluation instance, where it is automatically deployed and evaluated.

5. SUMMARY

Creating fully reproducible and comparable experiments in information retrieval is highly desirable, and various researchers have pointed out that advances in the state of the art in this field are difficult to account without such an achievement. A software service that meets this challenge and that is accepted within the research community must provide features such as local instantiation, web dissemination, platform independence, result retrieval, and peer to peer collaboration. The TIRA platform addresses these goals as a new web service to organize and operationalize specific programmable tasks runnable on the command line. Recently, TIRA has been deployed "in the wild" for the PAN series of international plagiarism detection competitions. Our preliminary findings are positive: even complex evaluations of software submissions can be easily managed, compared, and published. Based on this experience we aim to further develop TIRA towards a convenient tool for the information retrieval community to conduct evaluation initiatives.

Acknowledgements

The related work and architecture description (Sections 2 and 3) are reproduced from our paper "TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments" [4].

References

[1] J. Allan, W. B. Croft, A. Moffat, and M. Sanderson. Frontiers, Challenges, and Opportunities for Information Retrieval: Report from SWIRL 2012 The Second Strategic Workshop on Information Retrieval in Lorne. *SIGIR Forum*, 46(1):2–32, May 2012.

[2] T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel. Improvements that don't add up: Ad-hoc Retrieval Results since 1998. In D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu, and J. J. Lin, editors, *Proceedings of the Eighteenth ACM Conference on Information and Knowledge Management*, pages 601–610, Hong Kong, China, Nov. 2009.

[3] T. Gollub, B. Stein, and S. Burrows. Ousting Ivory Tower Research: Towards a Web Framework for Providing Experiments as a Service. In B. Hersh, J. Callan, Y. Maarek, and M. Sanderson, editors, *Proceedings of the Thirty-Fifth International ACM Conference on Research and Development in Information Retrieval (to appear)*, Aug. 2012.

[4] T. Gollub, B. Stein, S. Burrows, and D. Hoppe. TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments. In A. M. Tjoa, S. Liddle, K.-D. Schewe, and X. Zhou, editors, *Proceedings of the Ninth International Workshop on Text-based Information Retrieval at DEXA (to appear)*, Sept. 2012.

[5] J. P. A. Ioannidis. Why Most Published Research Findings Are False. *PLoS Medicine*, 2(8):696–701, Aug. 2005.

[6] M. Potthast. *Technologies for Reusing Text from the Web*. PhD Thesis, Bauhaus-Universität Weimar, Dec. 2011.