Continuous Integration for Reproducible Shared Tasks with `TIRA.io`

Maik Fröbe,[1] Matti Wiegmann,[2] Nikolay Kolyada,[2] Bastian Grahm,[3] Theresa Elstner,[3] Frank Loebe,[3] Matthias Hagen,[1] Benno Stein,[2] Martin Potthast[3]

[1] Friedrich-Schiller-Universität Jena
[2] Bauhaus-Universität Weimar
[3] Leipzig University

Abstract A major obstacle to the long-term impact of most shared tasks is their lack of reproducibility. Typically, only the test collections and the papers of the organizers and participants are published. Third parties must independently replicate participants' software if they want to evaluate state-of-the-art approaches for a task in their own experiments. The tools that have been developed to collect software from participants in shared tasks only partially verify their reliability at the time of submission and over the long term, and do not facilitate third parties to reproduce them later. We overhauled the TIRA Integrated Research Architecture to solve all these problems. The new version simplifies task setup for organizers and software submission for participants, scales from a local computer to the cloud, supports on-demand resource allocation up to parallel CPU and GPU processing, and enables export for local reproduction with a few lines of code. This is achieved by implementing the TIRA protocol with an industry-standard continuous integration and deployment (CI/CD) pipeline using Git, Docker, and Kubernetes.[4]

Keywords: Evaluation · Shared Tasks · Reproducibility

1 Introduction

A shared task is a collaborative laboratory experiment to evaluate state-of-the-art computational solutions to a problem, the task. A reproducible shared task collects the resources needed by third parties to reproduce the evaluation results. Reproducibility is only guaranteed if the datasets of the shared task and the software of the individual participants are available. However, most participants in shared tasks do not publish their software, and most organizers lack the time and tools to do so. Therefore, shared task results are typically difficult to reproduce by third parties after the completion of a shared task.

[4]Screencast for Reproducibility:	https://www.tira.io/t/post-hoc-experimentation
Screencast for Participants:	https://www.tira.io/t/screencast-for-participants
Screencast for Organizers:	https://www.tira.io/t/screencast-for-organizers

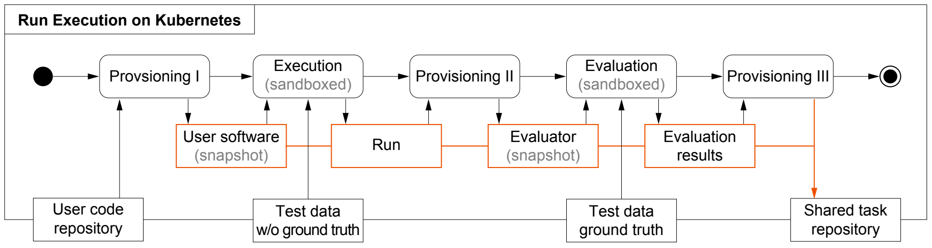

Figure 1. UML activity diagram of control flow (rounded boxes) and data flow (rectangular boxes). All run artifacts are persisted in the shared task repository by the three provisioning steps (highlighted arrow). The software and the evaluator are "untrusted" software and therefore executed in a sandbox without network access.

In information retrieval, many initiatives have been launched to promote the reproducibility of experiments and software. These include, among others, the reproducibility and resource tracks at the IR conferences, the OSIRRC workshop [2], and the CENTRE lab held in close succession at CLEF [4], TREC [13], and NTCIR [12]. Moreover, the ACM SIGIR Artifact Badging Board [3] awards badges of honor to especially reproducible papers. With respect to reproducible experiments and shared tasks, a requirements catalog has been developed at the Evaluation-as-a-Service (EaaS) workshop [7], drawing inspiration from existing tools [5, 6, 14, 15], as a guide to future ones [1, 8, 10, 11, 16].

In this paper, we present a new approach to make shared tasks reproducible based on continuous integration and deployment (CI/CD), a widely adopted industry standard in software engineering. In the context of research software engineering for shared tasks, this means the reproducible evaluation of every software version (CI, see Section 2), and the automatic archival of every evaluated software version in a centralized shared task repository (CD, see Section 3). The new approach is built on modern, widely adopted industry-standard tools including GitHub and/or GitLab, Docker, and the cluster computing framework Kubernetes. This minimizes the overhead for organizers and participants alike and enables immediate reproduction, all with just a few lines of code.

2 Snapshotting Participants Software on Every Run

Reproducibility is always a tradeoff between costs and benefits: perfect reproducibility in the last extremity may require the persistence of the hardware on which a software has been executed. Implementing a practical form of reproducibility, however, already is quite a challenge. TIRA initially persisted software by snapshotting virtual machines [5], which was prone to scaling errors due to its implementation in VirtualBox which required costly manual intervention. The recent progress in DevOps, continuous integration, and deployment (which are are well integrated with Git) provides all components to automate reproducible shared tasks using mature and cloud-native tools. Consequently, our new backend for TIRA is cloud-native, built on a privately hosted Gitlab and its container registry (12.4 PB HDD storage) and a Kubernetes cluster (1 620 CPU cores, 25.4 TB RAM, 24 GeForce GTX 1080 GPUs).

Figure 1 overviews TIRA's new workflow for shared tasks, which is centered around a shared tasks' Git repository and uses the tools integrated into Git platforms: (1) a user code repository with a dedicated container registry to which participants upload Docker images, (2) continuous integration runners that execute the five stages of the pipeline, (3) Kubernetes to orchestrate (provision, scale, and distribute) the runners in the cluster, and (4) a storage platform that provides access to the datasets.

Initially, users upload their software (in a Docker image) to their private container registry in their private code repository, maintained on the Git platform. Access is granted at the time of sign-up to TIRA via an autogenerated authentication token. Afterwards, users can add their software to TIRA by specifying the to-be-executed command for the image. Every software added this way is immutable: the command and the docker image can not be changed in retrospect (participants can upload multiple pieces of software and we do not set a limit). The container registry has a reduced memory footprint compared to virtual machines. Moreover, it is private during the task (only the user and TIRA have access) to avoid premature publication [9].

The workflow to execute and evaluate a software is specified via the declarative continuous integration API within the Git repository of the shared task, which also contains a registry for the evaluator images. The workflow is triggered with a special commit in the shared task repository (indicating the software to execute; performed via the TIRA web page or CLI) and consists of five steps:

1. *Provsioning I*: Prepares the local environment by branching and cloning the shared task's repository and copying the test data into the local environment (all operations are trusted).
2. *Execution*: In the prepared local environment containing the test data, this step sandboxes and then executes the user software to produce its output as 'Run' file(s). Sandboxing ensures that the (untrusted) user software can not leak test data by cutting the internet connection (using egress and ingress rules in Kubernetes), which enforces blind evaluation, and a software developed to sufficient maturity to run unattended in its Docker image (i.e., the software can not load models or code during its execution).
3. *Provisioning II*: Persists the run files and logs. Copies the test truth to the local environment for the evaluation step (all operations are trusted).
4. *Evaluation*: In the prepared environment containing the run files and the test truth, sandbox and then executes the evaluator to produce the evaluation results. The evaluator is not trusted because it is external software.
5. *Provisioning III*: Persists evaluation results and logs and merges the branch for this software execution into the main branch (all operations are trusted).

This workflow ensures that only the data of successfully executed and evaluated software is in the main branch of the Git repository of the shared task. Branches indicate queued or running software. Since the software itself is immutable, every software is snapshotted on every run.

```
import tira
df = tira.load_data('<dataset-name>')
# df can be manipulated for ablation/replicability/reproducibility studies
predictions, evaluation = tira.run(
    '<task-name>/<user-name>/<software-name>',
    data=df, evaluate='<evaluator-name>'
)
```

Listing 1: The software `<software-name>` by user `<user-name>` submitted in the `<task-name>` shared task is executed and evaluated on a pandas DataFrame `df` of dataset `<dataset-name>`. Demo available at tira.io/t/post-hoc-experimentation.

3 Repeat, Replicate, and Reproduce in One Line of Code

By way of continuous deployment (CD), our new version of TIRA delivers to organizers a self-contained Git repository that contains all shared task artifacts, ready for immediate publication. It includes all datasets, runs, evaluation results, logs, metadata, and software snapshots. This "shared task repository" also includes utility scripts to execute all software submitted to the shared task on the existing datasets, but also on different ones as long as the their formatting is the same. Listing 1 exemplifies this by loading a dataset (might be new or supplementary) into a pandas DataFrame that subsequently serves as input to a software submitted to the shared task while the predictions are directly evaluated. The utility script `tira` that enables this reproduction is in the repository; Docker and Python 3 are the only external dependencies. During archival of the shared task, all pieces of software are published at Docker Hub so that reproductions load them ad-hoc (moreover, TIRA maintains a local archive).

The shared task repository published in retrospect serves as a natural entry point to a wide range of follow-up studies. All researchers can fork this repository and contribute, increasing the impact of shared tasks even more as more supplementary material becomes available (e.g., datasets, evaluations, ablations, software, etc.). None of this was possible with the old version of TIRA, or any other related tool to support more reproducible experiments.

4 Conclusion

Our new version of TIRA enables reproducible shared tasks with software submissions in a cloud-native environment and is currently used in two shared tasks at SemEval 2023. The cloud-native orchestration reduces the efforts for organizing shared tasks with software submissions, so we plan to advertise TIRA further to get more shared tasks on the platform, enabling many post-hoc experiments that further foster software submissions in shared tasks.

For future work, we plan to port our pipeline to support more and also proprietary providers (GitHub, AWS/Azure) to make TIRA more accessible. We further aim at one-click deployments that use private repositories in GitHub or (self-hosted instances of) GitLab as the backend for shared tasks.

Bibliography

[1] Breuer, T., Schaer, P., Tavakolpoursaleh, N., Schaible, J., Wolff, B., Müller, B.: STELLA: Towards a Framework for the Reproducibility of Online Search Experiments. In: Clancy, R., Ferro, N., Hauff, C., Lin, J., Sakai, T., Wu, Z.Z. (eds.) Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019, CEUR Workshop Proceedings, vol. 2409, pp. 8–11, CEUR-WS.org (2019)

[2] Clancy, R., Ferro, N., Hauff, C., Lin, J., Sakai, T., Wu, Z.Z.: The SIGIR 2019 Open-Source IR Replicability Challenge (OSIRRC 2019). In: Piwowarski, B., Chevalier, M., Gaussier, É., Maarek, Y., Nie, J., Scholer, F. (eds.) Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019, pp. 1432–1434, ACM (2019)

[3] Ferro, N., Kelly, D.: SIGIR Initiative to Implement ACM Artifact Review and Badging. SIGIR Forum **52**(1), 4–10 (2018)

[4] Ferro, N., Maistro, M., Sakai, T., Soboroff, I.: Overview of CENTRE@CLEF 2018: A First Tale in the Systematic Reproducibility Realm. In: CLEF, Lecture Notes in Computer Science, vol. 11018, pp. 239–246, Springer (2018)

[5] Gollub, T., Potthast, M., Beyer, A., Busse, M., Rangel, F., Rosso, P., Stamatatos, E., Stein, B.: Recent Trends in Digital Text Forensics and its Evaluation. In: Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.) Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 4th International Conference of the CLEF Initiative (CLEF 2013), pp. 282–302, Springer, Berlin Heidelberg New York (Sep 2013), ISBN 978-3-642-40801-4, ISSN 0302-9743

[6] Hopfgartner, F., Brodt, T., Kille, J.S.B., Lommatzsch, A., Larson, M.A., Turrin, R., Serény, A.: Benchmarking News Recommendations: The CLEF NewsREEL Use Case. SIGIR Forum **49**(2), 129–136 (2015)

[7] Hopfgartner, F., Hanbury, A., Müller, H., Eggel, I., Balog, K., Brodt, T., Cormack, G.V., Lin, J., Kalpathy-Cramer, J., Kando, N., Kato, M.P., Krithara, A., Gollub, T., Potthast, M., Viegas, E., Mercer, S.: Evaluation-as-a-Service for the Computational Sciences: Overview and Outlook. ACM J. Data Inf. Qual. **10**(4), 15:1–15:32 (2018)

[8] Jagerman, R., Balog, K., de Rijke, M.: OpenSearch: Lessons Learned from an Online Evaluation Campaign. ACM J. Data Inf. Qual. **10**(3), 13:1–13:15 (2018)

[9] Lin, J., Campos, D., Craswell, N., Mitra, B., Yilmaz, E.: Fostering Coopetition While Plugging Leaks: The Design and Implementation of the MS MARCO Leaderboards. In: Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Culpepper, J.S., Kazai, G. (eds.) SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022, pp. 2939–2948, ACM (2022)

[10] Pavao, A., Guyon, I., Letournel, A.C., Baró, X., Escalante, H., Escalera, S., Thomas, T., Xu, Z.: CodaLab Competitions: An Open Source Platform to Organize Scientific Challenges. Tech. rep., Université Paris-Saclay, FRA. (2022)

[11] Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA integrated research architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF, The Information Retrieval Series, vol. 41, pp. 123–160, Springer (2019)

[12] Sakai, T., Ferro, N., Soboroff, I., Zeng, Z., Xiao, P., Maistro, M.: Overview of the NTCIR-14 CENTRE Task. In: Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies. Tokyo, Japan (2019)

[13] Soboroff, I., Ferro, N., Sakai, T.: Overview of the TREC 2018 CENTRE Track. In: The Twenty-Seventh Text REtrieval Conference Proceedings (TREC 2018) (2018)

[14] Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M.R., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., Almirantis, Y., Pavlopoulos, J., Baskiotis, N., Gallinari, P., Artières, T., Ngomo, A.N., Heino, N., Gaussier, É., Barrio-Alvers, L., Schroeder, M., Androutsopoulos, I., Paliouras, G.: An Overview of the BIOASQ Large-scale Biomedical Semantic Indexing and Question Answering Competition. BMC Bioinform. **16**, 138:1–138:28 (2015)

[15] Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: Networked Science in Machine Learning. SIGKDD Explor. **15**(2), 49–60 (2013)

[16] Yadav, D., Jain, R., Agrawal, H., Chattopadhyay, P., Singh, T., Jain, A., Singh, S., Lee, S., Batra, D.: EvalAI: Towards Better Evaluation Systems for AI Agents. CoRR **abs/1902.03570** (2019)