A Living Lab Architecture for Reproducible Shared Task Experimentation

Timo Breuer, Philipp Schaer

TH Köln – University of Applied Sciences, Germany
{timo.breuer, philipp.schaer}@th-koeln.de

Abstract

No existing evaluation infrastructure for shared tasks currently supports both reproducible on- and offline experiments. In this work, we present an architecture that ties together both types of experiments with a focus on reproducibility. The readers are provided with a technical description of the infrastructure and details of how to contribute their own experiments to upcoming evaluation tasks.

Keywords: reproducibility; evaluation infrastructure; shared tasks

1 Introduction

Experimental evaluation has a long history in Information Retrieval (IR) and Recommender System (RecSys) research and is the main driver of innovations and scientific progress in these fields. Shared task initiatives bring together community efforts and offer a platform for sharing and comparing different ideas and approaches. Especially in the field of IR, most experiments are based on test collections (Sanderson, 2010) with topical relevance judgments. Since pertinent relevance decisions are not covered, we can consider these in vitro approaches as *offline* experiments.

Very often, evaluations are solely based on system-oriented measures, and *online* experiments with real users are neglected. Opposed to this, Interactive Information Retrieval (IIR) experiments have a user-oriented focus (Kelly, 2007). Even though user-related aspects can be investigated in a very controlled manner, these experiments are costly, and thus are usually conducted on a smaller scale. As a compromise, session logs from online field experi-

ments allow us to include user interactions (as part of the evaluations) on a larger scale at the price of control over user characteristics.

A recent survey on shared task platforms shows, none of existing infrastructures ties together both worlds of on- and offline evaluations and additionally guarantees reproducible experiments (Schaible et al., 2020). In this sense, we introduce a novel architecture that complements the portfolio of the existing evaluation infrastructures by making online as well as offline evaluations fully reproducible. In the following, we broadly outline this infrastructure for which an overview is provided in Figure 1.

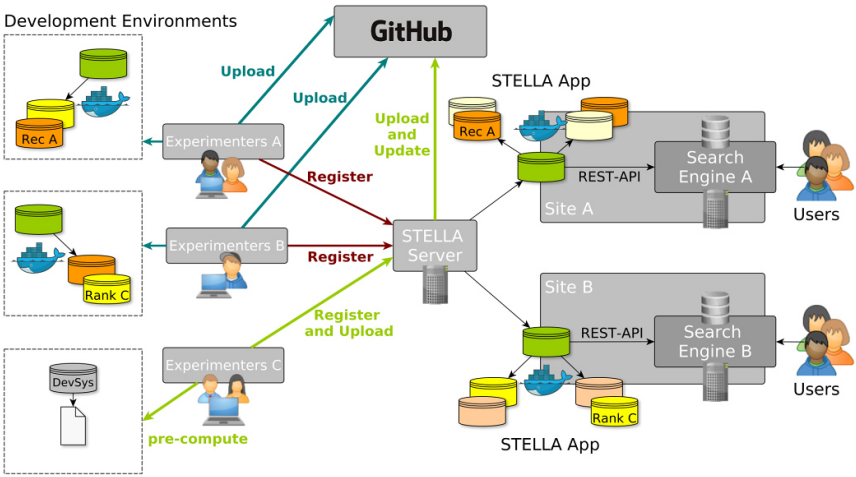

Fig. 1 Overview of the architecture

Our framework mainly relies on Docker and its containerization technology. An additional component is Git (or GitHub) as an attempt to make the whole infrastructure as transparent as possible. This is within the spirit of OpenScience and lets us focus on versioning, openness and reproducibility of runs and systems.

Experimenters provide systems with retrieval and recommendation algorithms and package them in Docker containers that are integrated into a multi container application (MCA). Multiple systems from possibly different experimenters are incorporated into the MCA. Search engine providers are referred to as sites. They provide access to data collections and, most importantly, user interaction data. Each site will deploy one instance of the MCA on their backend servers. Via a REST-API the users' queries can be conducted to the experimental systems. In return, the rankings and recom-

mendations are sent back. Sites will then log user interactions and send this feedback to the MCA. Eventually, the MCA sends the user interaction data to the central server where it is stored and can be used for further analysis, training, and optimization of the experimental systems.

The remainder is organized as follows. At first, we introduce each main component of our infrastructure and its functionalities, starting with the container, and continuing with the MCA and the central server. Afterward, we revise existing work and previous attempts to provide newcomers with the current state-of-the-art in the field of living lab experimentation. Finally, we conclude and end with future work.

2 The micro-services

As mentioned before, experimenters contribute their IR and RecSys algorithms as micro-services (or containers) that are integrated into the MCA. At the current state, the infrastructure provides two ways of contributing the corresponding systems or their outputs. In the following, we outline both to give the reader an impression of which one fits best to their evaluation practices.

2.1 Pre-computed results

Instead of submitting the entire IR or RecSys application, only its results are contributed for experimentation as illustrated in Figure 2. Like in previous living lab attempts (Jagerman et al., 2018), the pre-computed results are restricted to a specific set of queries that is extracted from the top-*k* results of query logs. For these head queries, rankings and recommendations must be derived from a given document collection (that can be retrieved after registration). In order to make the transition from offline to online evaluations as smooth as possible, these results are contributed with TREC run file syntax as illustrated below.

```
<qid> <Q0> <docid> <rank> <score> <identifier>
```

Run file syntax used for precomputed results

Each line contains the numeric query identifier (<qid>), a string identifying the document (<docid>), an increasing rank number (<rank>), the corresponding score (<score>), and the tag chosen by the experimenters (<identifier>). The files have to be uploaded to the central server either with the help of HTTP requests (via the REST-API) or via the user interface. The infrastructure service will automatically prepare the uploaded files for integration into the MCA after they have been checked for validity and consistency.

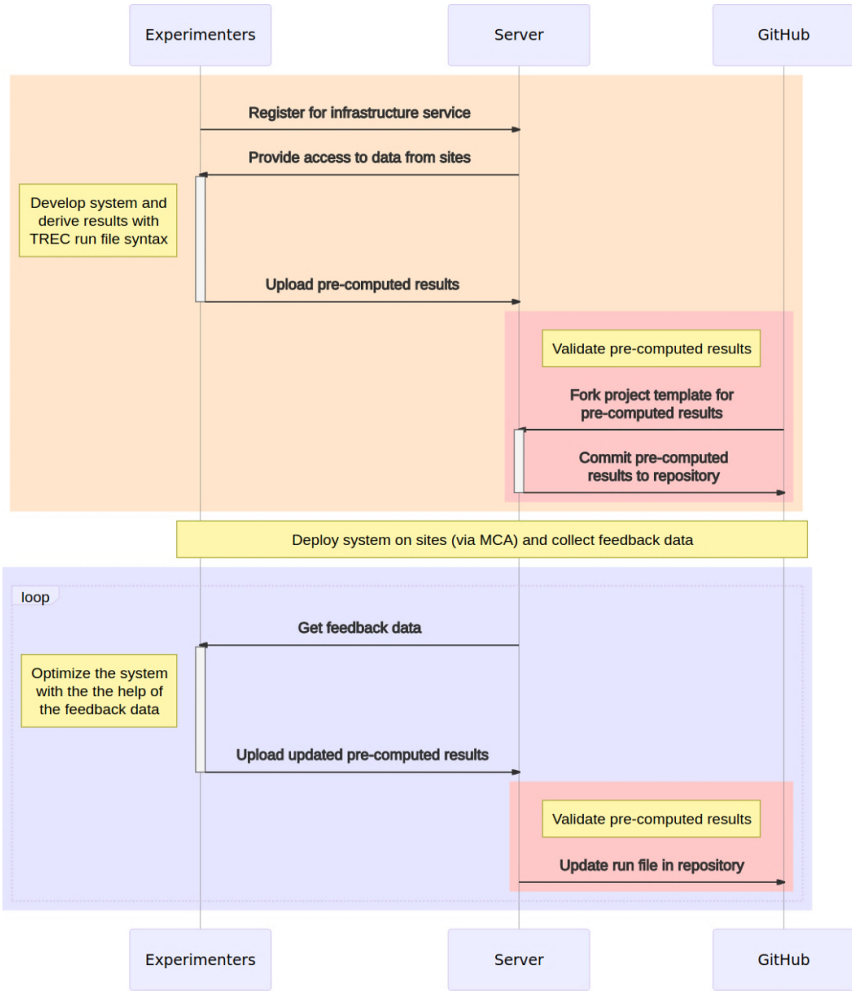

Fig. 2 Sequence diagram for contributing precomputed results

More specifically, a new micro-service is automatically set up and integrated into the MCA (cf. to the red boxes in Fig. 2). Furthermore, this submission type makes it possible to directly transfer over results from purely offline to living lab environments, facilitating the comparison of system-oriented evaluation measures to those inferred from, e.g., click feedback.

2.2 Encapsulated systems

Alternatively, experimenters can choose to submit the entire IR or RecSys application instead of systems' outputs only. In this case, the applications are contributed as individual micro-services, as illustrated in Figure 3.

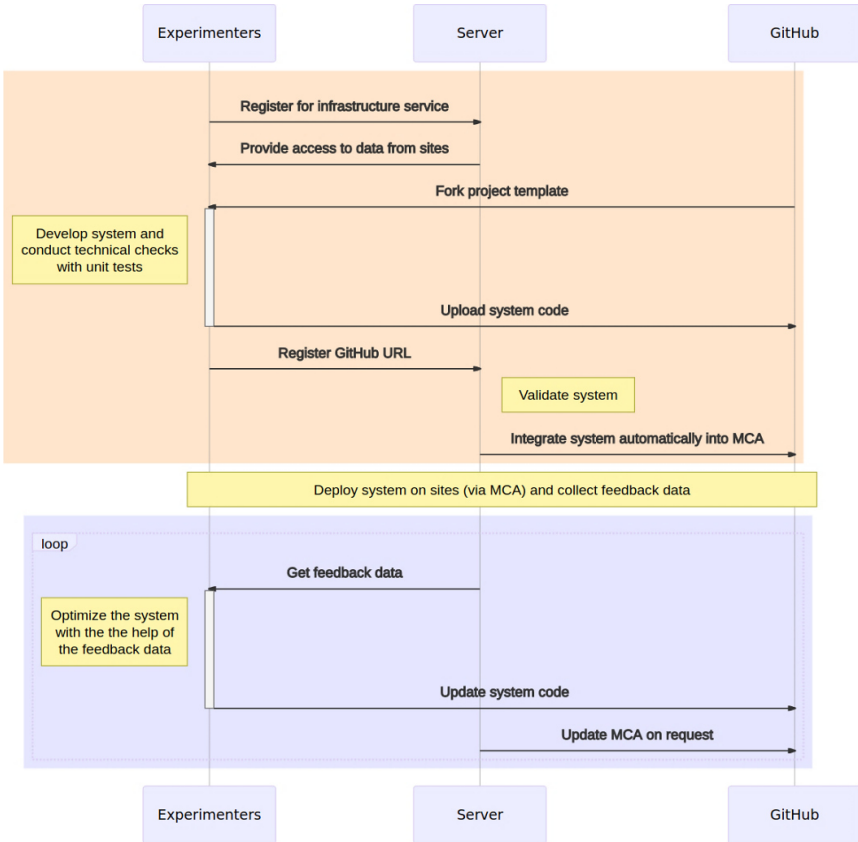

Fig. 3 Sequence diagram for contributing encapsulated systems

A project template in the form of a minimal Docker application is provided for the sake of compatibility between experimental systems and the MCA. Experimenters adapt these templates when integrating their applications. As a starting point, the project templates provide a minimal Python application based on the Flask web framework. But, of course, experimenters are not restricted to use Python at all and can modify the template as they see fit with the only requirement that resulting Docker containers respond to the defined REST endpoints correctly. These endpoints are then used by the MCA to index data or retrieve rankings and recommendations. An example of how to retrieve a ranking containing the first five results on the first page for the query *vaccine* is given below.

```
GET container/ranking?query=vaccine&page=1&rpp=5
```

REST endpoint for retrieving rankings from a single container

In comparison to pre-computed results, these micro-services are more comprehensive in the sense that their responses are not limited to pre-selected queries (or seed documents) only when rankings or recommendations are dynamically derived *on-the-fly*. The corresponding project template can be found in a public GitHub repository.[1]

3 The multi container application

Each IR or RecSys application is a micro-service integrated into the MCA. By using Docker as a packaging tool, it is assured that all systems are reproducible and set up as intended, and likewise, the deployment effort is reduced to a minimum. Having a reproducible system evaluated across different domains may help to gain insight in terms of generalizability.

The MCA is the entrance for site providers to the infrastructure with a broker-like role. Sites communicate with the MCA (by a REST-API), mainly for retrieving ranking and recommendations and sending user feedback. When a user enters a query into the user interface, the site conducts this query to the MCA and receives a JSON-formatted response like it is exemplified below.

1 https://github.com/stella-project/stella-micro-template

```
{
    "body":{
        "1":{ "docid":"M27622217", "type":"BASE" },
        "2":{ "docid":"M27251231", "type":"EXP"  },
        "3":{ "docid":"M27692969", "type":"BASE" },
        "4":{ "docid":"M26350569", "type":"EXP"  },
        "5":{ "docid":"M26715777", "type":"EXP"  }
    },
    "header":{
        "container":{
            "base":"rank_base",
            "exp":"rank_elastic"
        },
        "page":0,
        "q":"vaccine",
        "rid":3,
        "rpp":5,
        "sid":1
    }
}
```


JSON-formatted ranking returned by the MCA

Especially for small- to mid-scale experiments, traditional A/B tests are not feasible due to limited amounts of feedback data (Hofman et al., 2016). As a solution, we rely on relative user preferences by comparing systems side-by-side. Radlinski et al. (2008) introduced the Team Draft Interleaving (TDI), which interlaces two result lists. In contrast to A/B tests, less feedback data is required to infer which system performs better.

In the given example, the baseline (BASE) and an experimental (EXP) system compose the ranking. If multiple experimental systems can deliver results for a query, the system having the lowest number of impressions contributes its results. At the current state, sites should log if and when there was a click on a document and send this information to the MCA. From there on, the MCA takes over and sends the logged feedback to the central server at regular intervals. While the MCA also offers the possibility to conduct traditional A/B tests, two things must be considered. First, it requires tremendously more feedback data to gain reliable (and significant) insights between two test groups (ibid.). Second, interleaved rankings can be used implicitly to control the overall retrieval and recommendation quality. When interlacing the result lists with outputs of established IR or RecSys algorithm, it can be guaranteed that at least half of the rankings or recommendations deliver reasonable results. In contrast, exposing site users with entirely experimental results (by subpar systems) may cause frustration during search and, at the worst, damage the site's reputation.

Technically, the MCA is built with docker-compose[2] – a build automation tool for multi container Docker applications. With the help of a YAML file, it is possible to define multiple micro-services and combine them into one application. Each micro-service is defined by a separate entry in the YAML file. These entries will be automatically added by the infrastructure service once the systems passed the validation. A central micro-service administrates all other micro-services and provides the RESTful interface. It distributes incoming queries to the experimental systems, receives the logged feedback data, and posts them to the central server of the infrastructure. The implementation of the MCA is available in a public GitHub repository.[3]

4 The server

The central server of the infrastructure serves four basic functionalities, including 1) the administration of the infrastructure users, 2) the dashboard service, 3) automated updates of the MCA, and 4) hosting the user feedback data. The underlying technology stack is given in Figure 4.

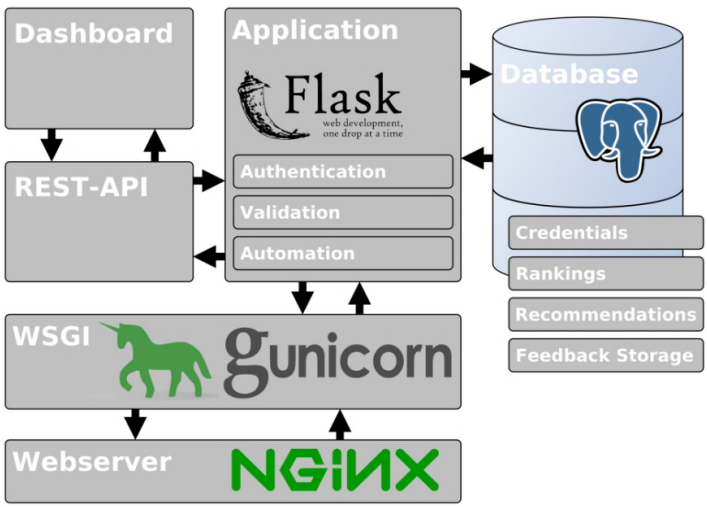

Fig. 4 Technology stack of the server

2 https://docs.docker.com/compose

3 https://github.com/stella-project/stella-app

Like the MCA and the template of the micro-services, the server application is implemented with Flask. It handles the authentication of users, validates submissions by experimenters and automates the build process of the MCA. Furthermore, it offers a RESTful API that is used by the MCA to post feedback data. Likewise, experimenters can use this API to retrieve feedback data which is stored in a PostgreSQL database.

After registration, new systems can be submitted by adding pointers to the corresponding GitHub-URL or by uploading pre-computed system results. Once systems' outputs have been exposed to site users, the dashboard service provides plots with the number of impressions and total clicks. Since we evaluate the systems' outputs with TDI, we can derive measures like wins, losses, and ties (Schuth et al., 2015). Figure 5 provides two examples of analytic tools that are provided by the dashboard service. These visualizations give first impressions of how the system performs in comparison to the baseline. Once enough data is available, experimenters can start the feedback loops that are illustrated in Figures 2 and 3. Clicks and other interactions can be used as points of reference for improving the ranking and recommender systems and the changes made can be compared to those of the previous evaluation phase. The implementation of the server is available in a public GitHub repository.[4]

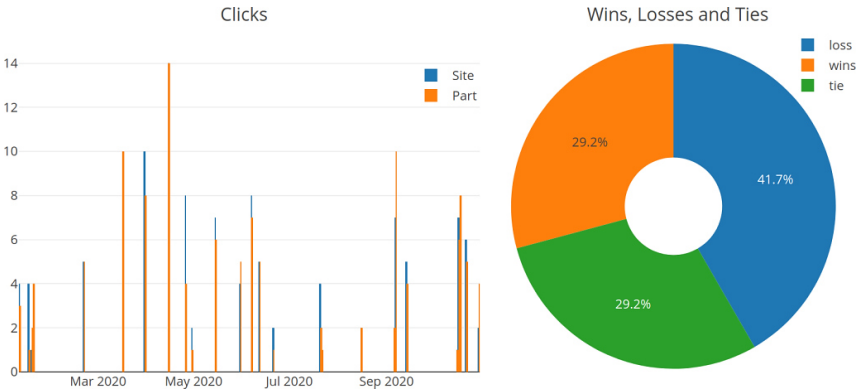

Fig. 5 Examples from the dashboard

4 https://github.com/stella-project/stella-server

5 Related work

Primarily, IR and RecSys methods should return results that are relevant to target users. While it is out of the scope of this work to elaborate on what constitutes relevance, researchers commonly agree that different types of relevance exist. Topical relevance judgments complement test collections (Sanderson, 2010) and simplify experimental evaluation but do not reflect the pertinent needs of specific searchers with different contexts.

Kelly's spectrum of IR experimentation contrasts experiments based on test collections with human-focused behavior studies at opposing ends (Kelly, 2007). IIR experiments can grasp very specific aspects of human behavior when interacting with search systems. However, these types of experiments are usually conducted on a small scale, and the recruitment of test subjects is costly. In the middle of her spectrum, Kelly locates log-based studies. They serve as a compromise between these two extremes of experiment designs, and one possible implementation of log-based studies are living labs (Kelly et al., 2009).

Living labs became part of IR conferences and workshops in 2014. Balog et al. (2014) introduced an API that can be used to retrieve experimental rankings for pre-selected queries, whereas Hopfgartner et al. (2014) introduced an infrastructure for real-time news recommendations. Both approaches share the idea of evaluating system performance with the help of user feedback that is provided in the form of clicks or other types of interaction with the presented search results. The living lab API for rankings was implemented in the LL4IR workshops (Schuth et al., 2015) as well as in the OpenSearch track (Jagerman et al., 2018), whereas the recommendation platform was used for the NEWSREEL campaign (Brodt & Hopfgartner, 2014).

While these attempts demonstrate the feasibility of living labs in IR and RecSys research, there remain many open questions and issues, e.g., regarding the reusability of such test collections that are augmented by user feedback data (Tan et al., 2017). More recent examples of living lab implementations are APONE (Marrero & Hauff, 2018), Macaw (Zamani & Crawell, 2020) and arXivDigest[5]. APONE is a living lab platform that is tailored to A/B tests as they are usually conducted in IIR studies. It is based on the PlanOut language developed by Facebook, which facilitates designing online

5 https://arxivdigest.org

field experiments by scripting them (Bakshy et al., 2014). Macaw has a focus on conversation information seeking, while its backend can either be fully algorithmic or set up as part of *wizard of oz* experiments. arXivDigest is a service that provides recommendations for research articles with the help of personalized email updates on recent publications based on interest profiles.

Recently, Schaible et al. (2020) provided a state-of-the-art overview on shared task evaluation platforms and found that there is no platform that combines both reproducible and online field experiments (as implemented by living labs). The presented infrastructure's underlying idea combines these two orthogonal research branches and can be framed with respect to larger reproducibility efforts in information retrieval (Breuer et al., 2019). Conceptually, the infrastructure takes account of the PRIMAD model (Ferro et al., 2016) that defines essential components of a reproducible experiment (including **P**latform, **R**esearch goal, **I**mplementation, **M**ethod, **A**ctor, **D**ata).

6 Conclusions and future work

In this work, we introduced a novel architecture for living lab experiments in IR and RecSys research. Living labs offer a new perspective when evaluating ranking and recommender systems. Conventional test collections merely support topical relevance, but not pertinent information needs of specific users. As a solution, feedback data, inferred from living lab experiments, complements system-oriented evaluations with new insights.

How does the proposed architecture differ from previous living labs? First, the way how systems are integrated into the infrastructure focuses on making the experiments reproducible. Furthermore, there is a frictionless transition from offline batch-style experiments to online environments. With the help of the introduced submission mechanism for pre-computed runs, system-oriented measures can be easily complemented by user feedback. Experimenters feed the datasets into their available ranking or recommender systems and simply upload the outputs for selected queries or target items to the central server of the infrastructure.

Previous living labs made use of pre-computed results for top-k queries that were available from a web-based service (Jagerman et al., 2018). When contributing full-fledged systems in Docker containers, there is no limitation to the top-k queries and, likewise, latencies should be reduced to a minimum,

since results do not need to be transferred over the web but will be retrieved from local servers at the sites. Furthermore, systems should be able to deliver rankings or recommendations *on-the-fly* when implementing them as micro-services. In comparison to pre-computed results, the outputs of these dockerized systems are more comprehensive since they should be able to deliver results for more than just pre-selected queries or target items. To lower the entrance barrier even further, we plan to provide project templates with dockerized versions of popular open-source search engines like Elasticsearch or Solr. Likewise, it is worth investigating the feasibility of a shared index for multiple experimental systems, as exemplified by the CIFF format (Lin et al., 2020).

At the current state, the provided datasets of the upcoming evaluation campaigns do not have topical relevance judgments. While it is possible to annotate specific rankings or recommendations after feedback data has been collected, it is a major challenge to infer the query logs' underlying information need and this may be a topic for future research. Queries are the users' verbalization and understanding of their information needs. In the outlined setup, we do not have any information about the users, thus no context knowledge. Different information needs may translate into the same query. Even if we are provided with the query, the resulting ranking, and the corresponding feedback data, it is not enough to reach any definitive conclusions about the information need. Similar problems occur for RecSys evaluations. In contrast to rankings, recommendations are not part of an active search process, but likewise we do not have any context knowledge and thus do not know what may have caused the users to click on a recommended item.

As part of our future evaluations, it is worth evaluating the infrastructure by comparing results from encapsulated systems to pre-computed results. Since experiments with pre-computations resemble older living lab attempts, we can treat them as a baseline when evaluating the new implementation of the living lab paradigm with the help of Docker.

In 2021, there is the Living Labs for Academic Search (LiLAS[6]) evaluation campaign at CLEF 2021 (Schaer et al., 2020). Interested readers are invited to contribute their IR and RecSys experiments and have them evaluated in the living lab environment that is introduced in this work. As part of the STELLA project, TH Köln, GESIS, and LIVIVO team up to offer lab partici-

6 https://clef-lilas.github.io

pants the possibility to expose their systems' results to users from the social and life sciences. A demo setup of the entire infrastructure can be found in a public GitHub repository.[7]

References

Bakshy, E., Eckles, D., & Bernstein, M. S. (2014). Designing and deploying online field experiments. In *Proceedings of the 23rd International Conference on World Wide Web – WWW '14*, pp. 283–292. https://doi.org/10.1145/2566486.2567967

Balog, K., Kelly, L., & Schuth, A. (2014). Head First: Living Labs for Ad-hoc Search Evaluation. *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management – CIKM '14*, pp. 1815–1818. https://doi.org/10.1145/2661829.2661962

Breuer, T., Schaer, P., Tavakolpoursaleh, N., Schaible, J., Wolff, B., & Müller, B. (2019). STELLA: Towards a Framework for the Reproducibility of Online Search Experiments. In *OSIRRC 2019, the Open-Source IR Replicability Challenge* (pp. 8–11). Aachen: RWTH Aachen. http://ceur-ws.org/Vol-2409/position01.pdf

Brodt, T., & Hopfgartner, F. (2014). Shedding light on a living lab: The CLEF NEWSREEL open recommendation platform. In *Proceedings of the 5th Information Interaction in Context Symposium – IIiX '14* (pp. 223–226). New York, NY: ACM. https://doi.org/ 10.1145/2637002.2637028

Ferro, N., Fuhr, N., Järvelin, K., Kando, N., Lippold, M., & Zobel, J. (2016). Increasing Reproducibility in IR: Findings from the Dagstuhl Seminar on "Reproducibility of Data-Oriented Experiments in e-Science". *SIGIR Forum, 50*(1), 68–82. https://doi.org/10.1145/2964797.2964808

Hofmann, K., Li, L., & Radlinski, F. (2016). Online Evaluation for Information Retrieval. *Foundations and Trends® in Information Retrieval, 10*(1), 1–117. https://doi.org/10.1561/1500000051

Hopfgartner, F., Kille, B., Lommatzsch, A., Plumbaum, T., Brodt, T., & Heintz, T. (2014). Benchmarking News Recommendations in a Living Lab. In E. Kanoulas, M. Lupu, P. Clough, M. Sanderson, M. Hall, A. Hanbury, & E. Toms (Eds.), *Information Access Evaluation. Multilinguality, Multimodality, and Interaction* (pp. 250–267). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-11382-1_21

7 https://github.com/stella-project/stella-search

Jagerman, R., Balog, K., & Rijke, M. D. (2018). OpenSearch: Lessons Learned from an Online Evaluation Campaign. *Journal of Data and Information Quality, 10*(3), 1–15. https://doi.org/10.1145/3239575

Kelly, Diane (2007). Methods for Evaluating Interactive Information Retrieval Systems with Users. *Foundations and Trends® in Information Retrieval, 3*(1–2), 1–224. https://doi.org/10.1561/1500000012

Kelly, D., Dumais, S., & Pedersen, J. O. (2009). Evaluation Challenges and Directions for Information-Seeking Support Systems. *Computer, 42*(3), 60–66. https://doi.org/10.1109/MC.2009.82

Lin, J., Mackenzie, J., Kamphuis, C., Macdonald, C., Mallia, A., Siedlaczek, M., Trotman, A., & de Vries, A. (2020). Supporting Interoperability Between Open-Source Search Engines with the Common Index File Format. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 2149–2152). New York, NY: ACM. https://doi.org/10.1145/3397271.3401404

Marrero, M., & Hauff, C. (2018). A/B Testing with APONE. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 1269–1272). New York, NY: ACM. https://doi.org/10.1145/3209978.3210164

Radlinski, F., Kurup, M., & Joachims, T. (2008). How does clickthrough data reflect retrieval quality? In *Proceeding of the 17th ACM Conference on Information and Knowledge Mining – CIKM '08* (pp. 43–52). New York, NY: ACM. https://doi.org/10.1145/1458082.1458092

Sanderson, M. (2010). Test Collection Based Evaluation of Information Retrieval Systems. *Foundations and Trends® in Information Retrieval, 4*(4), 247–375. https://doi.org/10.1561/1500000009

Schaer, P., Schaible, J., & Garcia Castro, L. J. (2020). Overview of LiLAS 2020 – Living Labs for Academic Search. In A. Arampatzis et al. (Eds.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction* (pp. 364–371). Cham: Springer International Publishing. https://doi.org/10.1007/ 978-3-030-58219-7_24

Schaible, J., Breuer, T., Tavakolpoursaleh, N., Müller, B., Wolff, B., & Schaer, P. (2020). Evaluation Infrastructures for Academic Shared Tasks: Requirements and Concept Design for Search and Recommendation Scenarios. *Datenbank-Spektrum, 20*(1), 29–36. https://doi.org/10.1007/s13222-020-00335-x

Schuth, A., Balog, K., & Kelly, L. (2015). Overview of the Living Labs for Information Retrieval Evaluation (LL4IR) CLEF Lab 2015. In J. Mothe et al. (Eds.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction* (pp. 484–496). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-24027-5_47

Tan, L., Baruah, G., & Lin, J. (2017). On the Reusability of "Living Labs" Test Collections: A Case Study of Real-Time Summarization. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 793–796). New York, NY: ACM. https://doi.org/10.1145/3077136.3080644

Zamani, H., & Craswell, N. (2020). Macaw: An Extensible Conversational Information Seeking Platform. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 2193–2196). New York, NY: ACM. https://doi.org/10.1145/3397271.3401415

In: T. Schmidt, C. Wolff (Eds.): Information between Data and Knowledge. Information Science and its Neighbors from Data Science to Digital Humanities. Proceedings of the 16th International Symposium of Information Science (ISI 2021), Regensburg, Germany, 8th–10th March 2021. Glückstadt: Verlag Werner Hülsbusch, pp. 348–362. DOI: doi.org/10.5283/epub.44953.