The Information Retrieval Experiment Platform

Maik Fröbe
Friedrich-Schiller-Universität Jena

Jan Heinrich Reimer
Friedrich-Schiller-Universität Jena

Sean MacAvaney
University of Glasgow

Niklas Deckers
Leipzig University and ScaDS.AI

Simon Reich
Leipzig University

Janek Bevendorff
Bauhaus-Universität Weimar

Benno Stein
Bauhaus-Universität Weimar

Matthias Hagen
Friedrich-Schiller-Universität Jena

Martin Potthast
Leipzig University and ScaDS.AI

ABSTRACT

We integrate `ir_datasets`, `ir_measures`, and PyTerrier with TIRA in the Information Retrieval Experiment Platform (TIREx) to promote more standardized, reproducible, scalable, and, if desired, even blinded retrieval experiments. Standardization is achieved when a retrieval approach implements PyTerrier's interfaces and the input and output of an experiment are compatible with `ir_datasets` and `ir_measures`. However, none of this is a must for reproducibility and scalability, as TIRA can run any dockerized software locally or remotely in a cloud-native execution environment. Version control and caching ensure efficient (re)execution. TIRA allows for blind evaluation when an experiment runs on a remote server / cloud not under the control of the experimenter. The test data and ground truth are then hidden from public access, and the retrieval software has to process them in a sandbox that prevents data leaks.

We currently host an instance of TIREx with 15 corpora (1.9 billion documents) on which 32 shared retrieval tasks are based, and with Docker images of 50 standard retrieval approaches on a mid-size cluster (1,620 CPU cores and 24 GPUs) on which automatically running and evaluating all approaches on all tasks ($50 \cdot 32 = 1{,}600$ runs) takes less than a week. This instance of TIREx is open for submissions and will be integrated with the IR Anthology.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; **Evaluation of retrieval results**.

KEYWORDS

Retrieval evaluation; Reproducibility; Shared tasks; TIREx

ACM Reference Format:

Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. The Information Retrieval Experiment Platform. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591888

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591888

1 INTRODUCTION

Research and development in information retrieval (IR) has been predominantly experimental. In its early days in the 1960s, the IR community saw the need to develop and validate experimental procedures, giving rise to the Cranfield paradigm [27], which became the de facto standard for shared tasks hosted at TREC [85] and many spin-off evaluations. Organizers of typical shared IR tasks provide a task description, a document corpus, and topics. Participants implement retrieval approaches for the task and run them on each topic to produce document rankings (a so-called "run"). The rankings are then usually submitted as files to the organizers who pool all runs, gather (reusable) relevance judgments for the pools, and calculate the evaluation scores [84]. Finally, participants describe their methodology and findings in a published "notebook" paper. This division of labor allowed the community to scale up collaborative laboratory experiments, especially at a time of limited bandwidths for data exchange, since run files occupy only a few kilobytes. With many research labs working independently on the same task, the community descends on a "wisdom of the crowd", while ensuring a rigorous comparative evaluation.

Despite the lasting success, this way of organizing shared tasks also has shortcomings. First, as with many other disciplines in computer science and beyond, the retrieval approach of a run described in a notebook paper might not be reproducible. There are well-documented cases where reproductions failed, despite putting much effort into it, even for approaches with diligently archived code repositories [1, 60]. Second, run submissions require that participants have access to the test topics, which has severe implications [43], such as informing (biasing) the research hypothesis or retrieval approach, unless researchers make a point of not looking at the topics, ever, during development. Third, it cannot be ruled out that current or future large language models have been trained, by mistake or deliberately, on publicly available test data, or that a usage warning stating not to use the data for training would go unnoticed.[1] In any case, the current best practices for shared tasks do not enforce "blinded experimentation"[2] with sufficient rigor, compared to other empirical disciplines.

To address all of these shortcomings, we have developed the IR Experiment Platform (TIREx; cf. Figure 1 for an overview). Available as open source,[3] a key feature of TIREx is the full integration

[1] Some form of leakage from MS MARCO [67] to the Flan-T5 prompting model [19] has already been observed: twitter.com/UnderdogGeek/status/1630983277363228672, twitter.com/macavaney/status/1649779164625481733.
[2] en.wikipedia.org/wiki/Blinded_experiment
[3] github.com/tira-io/ir-experiment-platform

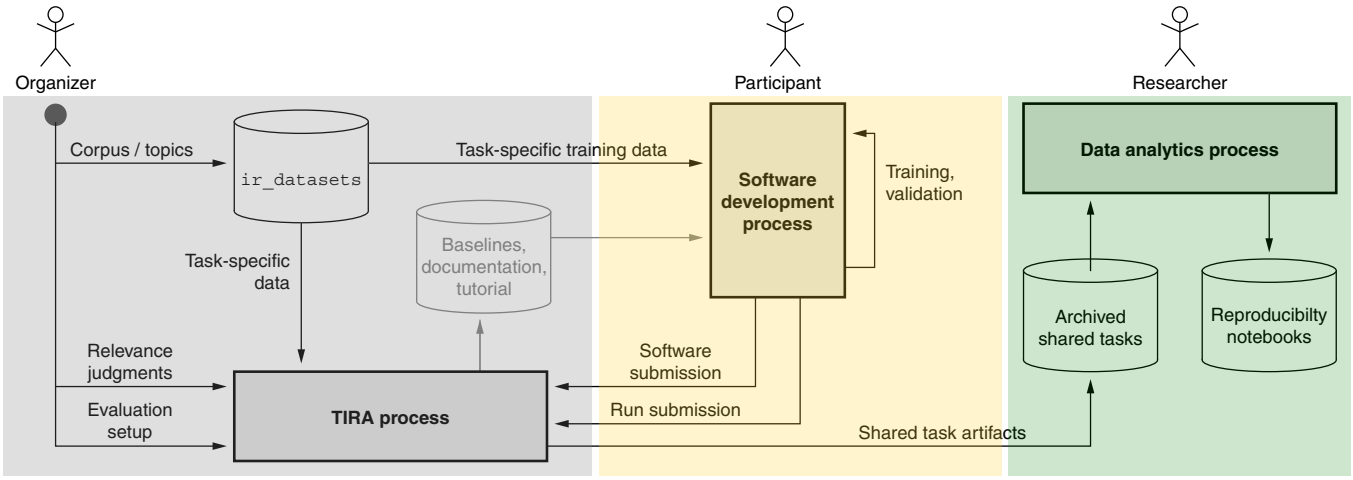

Figure 1: Overview of typical shared task-like IR experiments and how the tools in TIREx support them.

of tools for working with IR data (`ir_datasets` [63]), for executing retrieval pipelines (PyTerrier [64]), and for evaluating IR systems (`ir_measures` [61]) with TIRA [41], a continuous integration platform for reproducible shared tasks and experiments. TIREx is designed to improve reproducibility via software submissions while keeping an experimenter's or task organizer's workload at a degree comparable to run file submissions.

On our Betaweb and Gammaweb clusters,[4] we have deployed an instance of TIREx that is open for software submissions and experiments. As a proof of concept, we have conducted a large-scale evaluation of 50 "standard" retrieval approaches on 32 shared retrieval tasks (based on 15 document corpora with a total of 1.9 billion documents). This 1,600-runs experiment was started by just clicking a button and finished unattended in less than a week—a substantial efficiency boost comes from having GPU cores as part of the platform to speed up neural IR approaches.

2 BACKGROUND AND RELATED WORK

We review ad hoc retrieval experiments in evaluation campaigns, common problems and pitfalls in IR research, best practices for leaderboards, existing reproducibility initiatives, and tools to support reproducibility. Insights from all these domains have influenced our implementation decisions for TIREx.

Ad hoc Retrieval Experiments in Evaluation Campaigns. Today's shared task-style experiments for ad hoc retrieval evolved from the Cranfield experiments [85]. In the 1960s, the Cranfield experiments [27, 28] were conducted on a corpus of 1,400 documents with complete relevance judgments for 225 topics. Since corpus sizes grew substantially, complete judgments became infeasible almost immediately thereafter [85]. The current practice at shared tasks in IR thus is to only assess the relevance of per-topic pools of the submitted systems' top-ranked documents [85]. Subsequent evaluations on the same corpus usually are based on the assumption that the pools are "essentially complete", i.e., unjudged documents that were not in the pool are non-relevant [85]. Although this completeness assumption is reasonable for tasks with a diverse set of submitted runs and that were pooled at high depth [90], recent observations suggest that scenarios with many relevant documents per query (e.g., corpora with many duplicates [87]) or with topics representing broad information needs [79] are rather problematic. Especially for shared tasks that do not attract diverse submissions, TIREx can help to produce a more diverse judgment pool, as a wide range of baseline retrieval systems is directly available and can be applied to any imported retrieval task.

Common Problems and Pitfalls in IR Research. Even though the current discussion about how to conduct IR experiments [42, 76, 97] includes some controversial points (e.g., whether MRR should be abandoned [42] or not [66, 76]), there is still a wide consensus in the IR community on many characteristics of "bad" or "good" experiments. For instance, it is rather undisputed that retrieval studies should be internally valid (conclusions must be supported by the data) and externally valid (repeating an experiment on different but similar data should yield similar observations) [44]. Still, external validity of IR experiments remains an open problem [43]. TIREx can help to further improve both: the internal validity via archiving all experiments and results on some corpus (e.g., to accurately correct for multiple hypothesis tests), and the external validity via simplifying to run a submitted software on different data.

Thakur et al. [79] attempted to address the external validity problem by combining diverse retrieval corpora in the BEIR benchmark for *en masse* evaluation. However, in practice, running an approach on all corpora in BEIR requires some effort, so that many studies still only report results for a selected subset (e.g., [11, 40, 45])—often even without clearly justifying the selection. In contrast, a software in TIREx can rather easily be evaluated against many / all corpora so that analyzing improvements and limitations of an approach on diverse data is not much effort.

An often criticized practice is that many IR studies compare a new approach against weak / "wrong" baselines (i.e., not the best or most reasonable previous approaches). Any improvements claimed

[4] https://webis.de/facilities.html#hardware

in such studies are not really meaningful [3, 57]. One reason for choosing a wrong baseline could be that neither the researchers nor the reviewers are actually aware of what previous approaches exist for a specific corpus since results are often scattered across multiple publications [57]. Centralized leaderboards that directly show the effectiveness of diverse approaches for a wide range of tasks would address this problem, but multiple efforts have failed so far [57]. In TIREx, we include many popular corpora and standard retrieval approaches right from the start so that the TIREx leaderboards can initially gain traction. The more shared tasks (but also researchers) then later employ TIREx for software submissions, the broader TIREx' coverage will get over time.

Maintaining Ongoing Leaderboards. Inspired by the observation that many IR studies do not compare a new approach against reasonable baselines (e.g., the most effective TREC runs) [3], Armstrong et al. [2] released EvaluateIR, a public leaderboard accepting run file submissions. Although the concept was highly valuable for the community in helping researchers and reviewers alike to select appropriate baselines, "EvaluateIR never gained traction, and a number of similar efforts following it have also floundered" [57].

While there is still no centralized general leaderboard for IR, certain task-specific leaderboards are quite popular. For instance, the leaderboard of the recent MIRACL Challenge [96] received 25 submissions within one week, and the MS MARCO leaderboard [58] has been popular for years. Maintaining such long-running leaderboards comes with some caveats, as they are conceptually turn-based games where every leaderboard submission might leak information from the test set [58]. Lin et al. [58] propose best practices, inspired by previous problems of the Netflix prize.[5] Most importantly, they note that, while submissions to the leaderboard are open, the retrieval results should not be public, nor should system descriptions or implementations, as this would potentially leak information from the test set and foster "uninteresting" approaches like ensembles of all the top submissions. With TIREx and its blind evaluation, organizers can choose to blind all submissions as long as they need to, with the ability to unblind approaches and submissions as they see fit, so that TIREx supports the best practices recommended by Lin et al. [58].

Reproducibility Initiatives in IR. Reproducibility is a major challenge in research. For instance, a survey among 1,576 researchers revealed that more than 50 % failed at least once to reproduce their own experiments [5]. The IR community makes substantial efforts to foster reproducibility. There are, for instance, dedicated reproducibility tracks at conferences[6] and dedicated reproducibility initiatives like OSIRRC [1, 20] or CENTRE [38, 39, 77, 78]. OSIRRC aims to produce archived versions of retrieval systems that are replicable, while CENTRE runs replicability and reproducibility challenges across IR evaluation campaigns. Lin and Zhang [60] looked at all the artifacts produced in the OSIRRC 2015 challenge [1] to verify which results are still replicable four years after their creation. Out of the seven systems that participated in the challenge, only the results of Terrier [69] were fully reproducible out of the box, while other systems could still be fixed by manual adjustments to the code. The main reasons for failure were that external dependencies could not be loaded anymore, or that platform dependencies changed (operating system with its packages). To mitigate the problem of changing platform dependencies, the follow-up iteration of OSIRRC [20] focused on Docker images that had to implement a strict specification (enforced by the companion tool "jig") that triggered the indexing and subsequent retrieval via Docker hooks. Even though 17 systems have been dockerized to follow the jig specification, the concept has not gained traction. By centering TIREx around shared tasks at the beginning, we hope that we can kick off and maintain the attention of the community. Furthermore, we believe that there are many retrieval scenarios that can not be encapsulated into the two-step index-then-retrieve pipeline that jig imposes (e.g., explicit relevance feedback). We thus reduce the TIREx requirements to a minimum: just Docker images in which commands are executed without Internet access on read-only mounted data.

Tooling for Reproducibility. Many tools have been developed to support shared tasks by reducing the workload of organizers and participants while increasing the reproducibility [17, 41, 51, 53, 80, 81, 93]. For instance, as documenting the metadata of experiments improves reproducibility [56], `ir_metadata` [16] simplifies the documentation of IR experiments according to the PRIMAD model [37] (platform, research goal, implementation, method, actor, data). There are also platforms that support organizing and running shared tasks, among which four are still active: CodaLab, EvalAI, STELLA, and TIRA.[7] They implement the so-called evaluation-as-a-service paradigm in the form of cloud-based web services for evaluations [52]. Of these four systems, STELLA and TIRA are hosted within universities, while CodaLab and EvalAI use Microsoft Azure and Amazon S3, respectively. We use TIRA for TIREx as it allows blinded evaluation and as it is based on (private) git repositories hosted in GitLab / GitHub to versionize shared tasks and to distribute the workloads via runners connected to the corresponding repositories. The computation can thus be done in the cloud but also on private machines or clusters. We substantially redevelop large parts of TIRA as part of TIREx so that it supports the current IR workflows like chaining multiple retrieval stages.

3 THE IR EXPERIMENT PLATFORM

We describe how we integrate `ir_datasets`, `ir_measures`, and PyTerrier into TIRA to create the IR Experiment Platform (TIREx) to foster shared-task-style IR experiments with software submissions. We expect that the central components of this platform, TIRA, and `ir_datasets`, will be available and maintained over the coming years (TIRA has been maintained and developed since 2012 [46], and `ir_datasets` gained much traction by the community in the last 2 years). Although running shared-task-style IR experiments in TIRA was possible before, it required substantial effort from task organizers and participants due to idiosyncrasies of how IR experiments are conducted (as compared to typical ML or NLP experiments). Specifically, IR experiments involve intermediate artifacts, such as built indexes, and retrieval systems often involve multi-stage "telescoping" pipelines. We solve these problems by treating multi-stage pipelines as first-class citizens of the platform and by

[5] http://www.netflixprize.com/

[6] Examples at ECIR 2023 and SIGIR 2023: ecir2023.org/calls/reproducibility.html and sigir.org/sigir2023/submit/call-for-reproducibility-track-papers/.

[7] https://codalab.org, https://eval.ai, https://stella-project.org, https://tira.io

incorporating popular IR tools for data access, indexing, retrieval, and evaluation. The following section describes how shared-task-style IR experiments are organized in our new platform, explaining the integrated tools' interaction and providing examples for implementing prominent retrieval approaches. Finally, we showcase how TIREx enables post-hoc replicability and reproducibility studies with declarative PyTerrier pipelines.

3.1 Experiments in the IR Experiment Platform

TIREx covers all steps to organize retrieval experiments, as exemplified in Figure 1, and is open for registration. Organizers import their data, with all previously submitted retrieval software as potential baselines. Participants subsequently submit approaches, either as software submission, or, if enabled, as run submission. All submissions may be documented with descriptions and metadata, and run submissions may be grouped to indicate when the same approach created run files for multiple retrieval tasks. After gathering all submissions, organizers upload the relevance judgments to evaluate all runs. Finally, organizers export the complete experiment repository with all (meta) data and submitted software, which enables subsequent replication and reproduction experiments.

To import data into TIREx, experiment organizers add their corpus and topics to `ir_datasets`. Integrating the organizer's data into `ir_datasets` can be in a non-public branch if the data is confidential. The submission system TIRA imports the dataset via a Docker image with the corresponding `ir_datasets` installation. Participants make software submissions as Docker images, and TIRA uses sandboxing (removing the internet connection from the running software) to improve reproducibility (i.e., ensuring the software is fully installed). The software can use additional data as input that participants uploaded via run uploads, which documents dependencies to possibly non-reproducible parts of submissions (e.g., manual query reformulations). Participants may use the existing starters for 4 frequently used IR research frameworks as the basis for development. The simplest starter implements BM25 retrieval using a few lines of declarative PyTerrier code in a Jupyter notebook.[8] The software submissions are executed on demand using a cloud-native execution environment with GitLab/GitHub CI/CD pipelines so organizers can add runners as needed. TIRA organizes and versions all aspects of the retrieval experiment in a dedicated git repository that can be exported and published. Those experiment archives are fully self-contained, allowing the stand-alone re-execution of archived approaches on the same or different data in PyTerrier pipelines for reproducibility. Altogether, this substantially enriches the assets resulting from experiments, allowing "always-on reproducible shared tasks" for the IR community.

3.2 Reproducible Shared Tasks with TIRA

TIRA is used since 2012 to organize shared tasks with software submissions, with PAN[9] and Touché[10] being two long-running tasks in TIRA hosted at CLEF [46, 71]. The first version of TIRA facilitated software submissions during shared tasks by providing participants access to virtual machines. We found that this did not scale and was prone to errors, making it very difficult for external researchers to re-execute software submitted to a shared task. Hence, TIRA was completely redeveloped based on industry-standard continuous integration and deployment (CI/CD) pipelines using Git, Docker, and Kubernetes [41]. In the new version of TIRA, participants upload their software, implemented in Docker images, to a private Docker registry (12.4 PB HDD storage) dedicated to their team, ensuring that different teams do not influence each other while the task is running, as their approaches remain private until after the task completed. For the on-demand execution, TIRA runs the software in a Kubernetes cluster (1,620 CPU cores, 25.4 TB RAM, 24 GeForce GTX 1080 GPUs) in sandbox mode. This new version of TIRA was first used in two large-scale NLP tasks hosted at SemEval 2023 that together had 170 registered teams, out of which 71 teams submitted results, yielding 647 runs produced by software submissions (covering the usual participation for shared tasks in IR). However, during the setup of the next iteration of the retrieval-oriented Touché task for CLEF 2023 [8], we recognized that TIRA still had severe shortcomings for IR tasks, substantially limiting its adoption. TIRA had no unified data access, and typical IR workflows were only realizable inefficiently or via workarounds (such as adding the index into the image, harming reusability and reproducibility). TIRA had no separation in full-rank or re-rank software, and it was impossible to modularize software into separate components with caching. For instance, full-rank retrieval would require any software to build the index from scratch, wasting many resources. Similarly, re-rank approaches would have to create their initial ranking, making the software more complex and prone to error while wasting computational resources and hindering reusability. For instance, task 2 of Touché retrieves against the ClueWeb22, which is made available in TIRA through ChatNoir [7], but retrieving the top-1000 results for 50 Touché [8–10] topics against ChatNoir took, for 5 repetitions, between 54 and 134 minutes (top-1000 searches often fail so that a client has to retry the requests). Blocking a GPU that re-rankers often require for such a long time would be wasting resources (more complex re-ranking software that employs parallelism is not an option as this is prone to error). To solve all these problems, we substantially expanded TIRA and redeveloped major parts to integrate `ir_datasets`, `ir_measures`, and PyTerrier.

3.3 Standardized Access with ir_datasets

The `ir_datasets` toolkit provides unified access to a wide range of corpora frequently used in IR experiments [63]. Processing topics and documents is possible via a single line of Python code. Further, it already serves as a common data layer in numerous IR research frameworks and tools (e.g., PyTerrier [64], Capreolus [95], OpenNIR [62], FlexNeuART [13], Patapsco [31], Experimaestro-IR [70]), and can be easily ingested by most others (e.g., Anserini [94], PISA [65]). The tool provides a standard interface to access over 200 distinct document corpora and over 500 topic sets, and is kept up-to-date (e.g., it includes most of the TREC 2022 tracks). We integrate `ir_datasets` into TIRA so that retrieval software may leverage all structured information, but also can just use default texts for documents and queries, enabling retrieval software to be applied to different data without adaptation. We integrate `ir_datasets` into TIRA via Docker images that can import complete corpora into

[8] https://github.com/tira-io/ir-experiment-platform#starter-for-pyterrier-in-jupyter
[9] https://pan.webis.de/
[10] https://touche.webis.de/

TIRA (for full-rank approaches), but also can create re-ranking files for any given run file (for re-ranking approaches). Within the configuration of an IR experiment in TIRA, organizers choose the `ir_datasets` Docker image and its configuration. We provide standard Docker images that organizers can use if their dataset is already available in `ir_datasets`. Naturally, task organizers can also provide an image with data sourced from elsewhere, e.g., if the organizers want to keep their data private to avoid leaking data. In the following, we first describe how we implement the "default text" feature that allows to easily re-use retrieval software in different retrieval tasks, and how we integrate `ir_datasets` into Docker images that run on-demand, ensuring the interchangeability and compatibility of arbitrary retrieval software in retrieval pipelines.

Re-Usable Retrieval-Methods with Default Texts. While some corpora provide only a single freeform text field for each document, others provide rich structural information and other metadata. For instance, the MS MARCO passage ranking corpus [35, 36, 67] provides a single text field, while Args.me [9, 10] contains structured premises, aspects, and other metadata. Similarly, some retrieval tasks include a single freeform text field for topics, while others provide multiple versions of the query and/or metadata for each topic. For instance, Antique [47] has a single text query field, while TREC Precision Medicine [74, 75] provides multiple fields.

For corpora and retrieval tasks that provide rich structure, we identify two use cases. The first is when a system is built-for-purpose, making use of the available task-specific structure. For instance, an argument retrieval system on the Args.me corpus may make special use of the premises of a document during retrieval, or a Precision Medicine system may choose to use the structure of the query to modify the relevance criteria. Providing all the available fields in the input data facilitates the construction of this style of specialized retrieval systems.

The second use case is when a system seeks to provide the capacity for general search, i.e., an algorithm that can be applied across a variety of search applications, rather than targeting one specific case. To facilitate this use case, we introduce a "default text" field for each topic and document into `ir_datasets`, enabling systems to represent each query and document as a single string, regardless of the retrieval task. Often there is a natural choice for a query or document's "default text". For instance, MEDLINE documents contain both a title and abstract; in this case, the default text is the concatenation of the two fields. Many retrieval tasks contain title, description, and narrative versions for topics. Since the "title" usually refers to the style of query that users would enter into search engines, we use the title as the default text in the cases where it is intended as the to-be-submitted query (after a manual review). Other cases are not as clear cut. In these cases, we make our best effort to faithfully represent the most important content of the document or topic, being open to pull requests by the community. We incorporate the default text fields into the `ir_datasets` package and flow the data through to TIREx. This allows for a clear and documented provenance of the fields from the source material.

Ensuring the Compatibility of Modularized Retrieval Stages. TIREx aims to support modularized retrieval pipelines where different stages can be exchanged without adopting the retrieval software. Therefore, TIRA allows two different types of retrieval software: (1) full-rank approaches with the complete corpus and the to-be-retrieved topics as input, and (2) re-rankers with a specific re-rank file as input that was automatically created by `ir_datasets` from any run file. This way, re-rankers can run on arbitrary previous stages, as their input always has the same structure. TIRA runs the configured `ir_datasets` image on-demand to create those re-rank files. Corpora that can not be downloaded from the Web (e.g., the ClueWeb or the Gov corpora) are mounted into the container. Some corpora available in TIREx (e.g., the ClueWeb or Gov corpora) require license agreements. As we have valid license agreements for those corpora, by default we let participants execute their software on those corpora but only return effectiveness scores of their approaches (the outputs are blinded by default).

Table 1 overviews the data format that the `ir_datasets` integration makes available to retrieval software. For full-rank software, the `ir_datasets` integration creates a `documents.jsonl.gz` file that contains the document identifier, the default text of the document, and all structured fields of the document and a file `topics.jsonl.gz` with the topics, both in JSON Lines format. For re-rankers, the `ir_datasets` integration creates, given an arbitrary run file obtained from the previous stage, a file `re-rank.jsonl.gz` where each entry contains to-be-re-ranked query-document pairs with the score and rank assigned by the previous stage. Re-rankers should then re-rank all documents into their output run file. Any reranker added to TIRA can re-rank the results of any other retrieval software that creates a run file, in which case the `ir_datasets` integration is executed before the re-ranker and the re-ranker only gets the `re-rank.jsonl.gz` file as input. Furthermore, the `ir_datasets` integration makes the `qrels.txt` available (only if the dataset already contains relevance judgments) to the evaluator specified by the organizer to automatically evaluate submitted retrieval approaches.

3.4 Evaluation with ir_measures

TIRA automatically evaluates all runs created by software submissions or uploads. We provide an `ir_measures` evaluator suitable for IR experiments. If no relevance judgments are available, this evaluator checks that the run file can be parsed and warns of potential pitfalls (e.g., score ties, NaN scores, empty result sets, unknown queries, scores contradicting the ranks, etc). With relevance judgments, the evaluator scores all specified measures as average over all queries and per query (suitable for significance tests).

3.5 Reproducible IR Pipelines with TIRA

To make frequent IR workflows efficient first-class citizens in TIREx, we redevelop and extend TIRA's ability to define and run modularized software that spans multiple, potentially different Docker images, and implement a separation of retrieval systems into full- and re-rank approaches. All software in TIRA is immutable. Hence, outputs of software components shared across different retrieval software (e.g., index creation) can be cached, making full-rank and re-rank software more efficient.

Modularized Software from Multiple Components. Retrieval software in TIRA can be composed of multiple software components forming a directed acyclic graph. Each component (full-rank or re-rank) is defined by its preceding components (none, one, or many), its Docker image, and the command that is executed within the

Table 1: Overview of what data TIRA makes available to full-rank and re-rank approaches. The 'Access' columns indicate the default accessibility to participants (P), organizers (O; can make data accessible as indicated by †), and unregistered users (U).

Type	Resource	Fields	Access			Example Entry
			P	O	U	
Full-Rank	documents.jsonl.gz	docno, text, original_document	✓	✓	✗†	{"docno": "8182161", "text": "Goldfish can grow up to 18 inches …", "original_document": {…}}
	topics.jsonl.gz	qid, query, original_topic	✓	✓	✗†	{"qid": "156493", "query": "do goldfish grow", "original_query": {…}}
Re-Rank	re-rank.jsonl.gz	qid, query, original_topic, docno, text, original_document score, rank	✓	✓	✗†	{"qid": "156493", "query": "do goldfish grow", "original_query": {…}, "docno": 8182161, "text": "Goldfish can grow up to 18 inches …", "original_document": {…}, "rank": 1, "score": 31.16}
Both	qrels.txt	topic, iteration, docno, relevance	✗†	✓	✗†	156493 Q0 8182161 2

Table 2: Overview of variables available for software in TIRA. The `$inputDataset` and `$outputDir` variables are always available, while `$inputRun` is only available for multi-component software depending on previous stages.

Variable	Availability	Description
`$inputDataset`	Always	Directory containing the input data.
`$outputDir`	Always	Directory with expected output data.
`$inputRun`	Multi-Comp.	Output(s) of previous stage(s).

Docker image. TIRA passes the input and output directories to each software via three variables explained in Table 2. The variable `$inputDataset` points to the directory that contains the input passed to the software (e.g., `documents.jsonl.gz` and `topics.jsonl.gz` for full-rank and `re-rank.jsonl.gz` for re-rank software). The variable `$inputRun` is only available for software composed of multiple components, pointing to a directory with all outputs of preceding components. The variable `$outputDir` specifies the location where TIRA expects all outputs. All three variables `$inputDataset`, `$inputRun`, and `$outputDir` can be used in the to-be-executed command but are also made available as environment variables.

Manual Input Components. Retrieval runs might depend on inputs not available in the data created by humans. In TREC-style evaluation campaigns, such runs are called manual submissions. For instance, user query variants, as employed on the ClueWebs [4] or Common Core [6], are examples for manual submissions: researchers read the topic description and narrative and formulate (multiple) queries they would submit against a search engine that then serves as additional input to some retrieval algorithm. TIREx supports this use case via run uploads. The uploaded files can be grouped and documented and can be configured as a preceding component in any software, identical to software components. Consequently, TIREx supports manual runs, but isolates the manual steps as much as possible to keep the overall software replicable.

Examples for Frequent Retrieval Pipelines. Figure 2 provides a conceptual overview of the data flow between subsequent software components in two frequent approaches. The upper software described in Figure 2 shows how full-rank approaches might first create an index of the corpus from which the second component retrieves. Therefore, the first component "Index Corpus" creates an index, denoted by `file-01` and `file-02`, that it stores in `$outputDir`. TIRA then makes the output of the first component available to

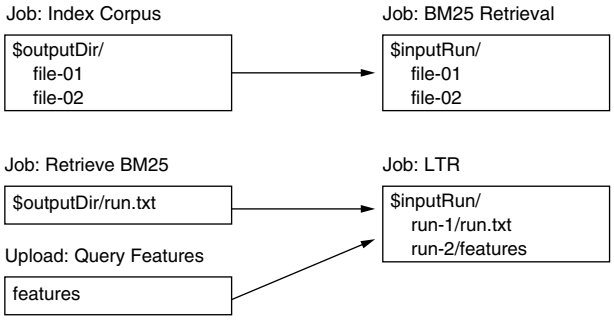

Figure 2: Data flow of two retrieval pipelines in TIRA. The upper retrieval pipeline creates an index so that the second stage retrieves from the index with BM25. The bottom retrieval pipeline uses a BM25 ranking and a manually uploaded file with query features as input for an LTR algorithm.

the subsequent software "BM25 Retrieval" inside `$inputRun`. This way, many different software components may depend on the same component, and we cache outputs to make pipelines more efficient. The lower software described in Figure 2 shows an example of a learning-to-ranking component that depends on a BM25 retrieval and a manual upload that provides query features. If a software component has more than one input, TIRA makes them available in the order in which they were defined. Figure 3 shows how retrieval software with multiple components can be defined within TIRA. All preceding software components and run uploads must exist so that the new software can add them as preceding components, as shown in Figure 3 that defines a software that first creates a PyTerrier index and then applies BM25 retrieval on this index. TIRA decouples the to-be-executed command from the Docker image to explicate that the same image can be used to produce different retrieval software (e.g., by switching parameters). In combination with multi-stage pipelines, this allows the efficient execution of a wide range of prominent retrieval pipelines (i.e., different retrieval models implemented in the same docker image) through caching.

Caching of Results. The fact that any retrieval software in TIRA is immutable allows us to build efficient pipelines by caching the outputs of software components. Although each component can be executed multiple times on the same dataset, subsequent stages get only the outputs of the first execution as input, and an output that

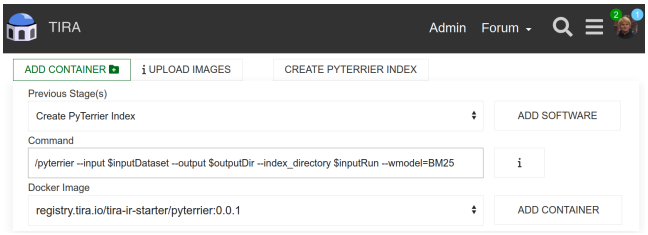

Figure 3: The definition of a full-rank retrieval software in TIRA that consists of two modularized components.

```
pipeline = tira.pt.retriever(
    '<task-name>/<user-name>/software',
    dataset='<dataset>'
)
advanced_pipeline = pipeline >> advanced_reranker
```

Listing 1: Full-rank retrieval from a complete corpus.

was used by some other component as input can not be deleted. Cleaning up is possible by first deleting runs that are not used as inputs. Programming errors in some components have to be fixed by adding a new software (the old software can be deleted when all runs are deleted), as all software is immutable. Altogether, retrieval pipelines in TIRA provide freedom for participants and allow to efficiently re-use shared components as they are only executed once and subsequently are served from the cache. The overall retrieval pipelines remain replicable, as the steps to produce a final run are fully tracked and versioned by TIRA in the experiment repository.

3.6 Reproducibility Pipelines with PyTerrier

After the experiment repository is exported and published by the organizers, the submitted retrieval approaches and the produced datasets, and run files can be re-used and applied to replicability and reproducibility studies. By default, TIRA keeps the test data private, publishing only the software submissions and run files, uploading the images to Dockerhub. After archival, the shared task repository and all possible follow-up studies are independent of TIRA and work stand-alone, as the archived repository is fully self-contained. In the following, we show the integration of archived TIRA repositories with PyTerrier for post-hoc experiments that can run after an experiment was completed and the data was published.[11]

Listing 1 shows how a full-rank approach submitted to TIRA can be reproduced with a declarative PyTerrier pipeline. The software submission is identified by `<task-name>/<user-name>/software`, i.e., it points to the software `software` submitted by team `<user-name>` to the shared task `<task-name>` and is applied to the specified dataset. Internally, this command downloads the required Docker images and runs them in their required order to obtain the results. This full-rank software can subsequently be re-ranked by any PyTerrier re-ranker, allowing for experiments that try to improve upon an original submission. The passed dataset can be one of the original experiments or any other dataset available in `ir_datasets`. Similar, Listing 2 shows how re-rankers can be used in post-hoc experiments by first retrieving results with BM25 which are then re-ranked with an approach submitted to some experiment in the platform.

[11] Examples available at: github.com/tira-io/ir-experiment-platform#reproducibility

```
bm25 = pt.BatchRetrieve(index, wmodel="BM25")
reranker = bm25 >> tira.pt.reranker(
    '<task-name>/<user-name>/<software>'
)
```

Listing 2: Run a re-ranker submitted as software to a task.

```
first_stage = tira.pt.from_submission(
    '<task-name>/<user-name>/<software>',
    dataset='<dataset>'
)
advanced_pipeline = first_stage >> advanced_reranker
```

Listing 3: Re-rank a run created by a software submission.

Listing 3 shows how run files resulting from some (software) submission can be loaded into PyTerrier. This `from_submission` method allows building upon submitted systems without having to re-run them (e.g., for re-rankers), also allowing organizers to build the judgment pools. Altogether, the PyTerrier integration allows easy replicability (if the dataset is the same as in the original experiment) and reproducibility experiments (if the dataset was not used in the original experiments) for full-rank and re-rank approaches.

4 EVALUATION AND PREDICTED IMPACT

We show the scalability and the potential impact of TIREx by importing 50 different retrieval approaches (covering all major paradigms) and 32 different retrieval tasks from 15 corpora with overall 1.9 billion documents. We run every retrieval software on all 32 tasks, yielding 1,600 runs. The submission to all those tasks and the resulting leaderboards are open.[12] We provide a case study analyzing what observations transfer between selected tasks with `repro_eval` [15]. Finally, we report on our experiences in hosting two large-scale NLP tasks with software submissions at SemEval 2023, concluding with a discussion on the predicted impact of the platform.

4.1 Initial Retrieval Experiments

Table 3 overviews the 15 corpora, each with 1 to 4 retrieval tasks and 1,400 to 1 billion documents, that we import into TIREx. All tasks come with dense judgments, typically created in a TREC-style shared task, covering diverse retrieval tasks.

Table 4 overviews the 50 different retrieval approaches that we imported into TIREx. We derived all retrieval approaches from 4 retrieval frameworks: BEIR [79], ChatNoir [7], PyGaggle [59], PyTerrier [64], and two PyTerrier plugins for duoT5 [72] and ColBERT [55]. From BEIR, we obtain 17 dense retrieval approaches (e.g., ANCE [92], DPR [54], TAS-B [50], etc.) by using different SBERT [73] models available in BEIR. ChatNoir is an Elasticsearch-based BM25F search engine hosting all three ClueWebs. ChatNoir can be accessed from within TIRA to realize retrieval approaches on huge corpora (we keep its REST-API consistent to ensure reproducibility). Out of PyGaggle, we include overall 8 variants of monoBERT [68] and monoT5 [72], including the monoT5 SOTA

[12] https://github.com/tira-io/ir-experiment-platform#submission

Table 3: The 15 corpora and the associated 32 retrieval tasks currently available in TIREx (submission possible).

Corpus			Associated Retrieval Tasks	
Name	Docs.	Size	Details	#
Args.me	0.4 m	8.3 GB	Touché 2020–2021 [9, 10]	2
Antique	0.4 m	90.0 MB	QA Benchmark [47]	1
ClueWeb09	1.0 b	4.0 TB	Web tracks 2009–2012 [22–25]	4
ClueWeb12	731.7 m	4.5 TB	Web tracks [29, 30], Touché [9, 10]	4
ClueWeb22B	200.0 m	6.8 TB	Touché 2023 [8] (ongoing)	1
CORD-19	0.2 m	7.1 GB	TREC-COVID [86, 91]	1
Cranfield	1,400	0.5 MB	Fully Judged Corpus [27, 28]	1
Disks4+5	0.5 m	602.5 GB	TREC-7/8 [88, 89], Robust04 [82, 83]	3
GOV	1.2 m	4.6 GB	Web tracks 2002–2004 [32–34]	3
GOV2	25.2 m	87.1 GB	TREC TB 2004–2006 [18, 21, 26]	3
MEDLINE	3.7 m	5.1 GB	TREC Genomics [48, 49], PM [74, 75]	4
MS MARCO	8.8 m	2.9 GB	Deep Learning 2019–2020 [35, 36]	2
NFCorpus	3,633	30.0 MB	Medical LTR Benchmark [12]	1
Vaswani	11,429	2.1 MB	Scientific Abstracts	1
WaPo	0.6 m	1.6 GB	TREC Core 2018	1
$\sum$ = 15 corpora	1.9 b	15.3 TB		32

Table 4: Overview of the retrieval frameworks and the 50 retrieval approaches imported into TIREx.

Framework	Type	Description	Approaches
BEIR [79]	Bi-encoder	Dense retrieval	17
ChatNoir [7]	BM25F	Elasticsearch cluster	1
ColBERT@PT [55]	Late interaction	PyTerrier plugin	1
DuoT5@PT [72]	Cross-encoder	Pairwise transformer	3
PyGaggle [59]	Cross-encoder	Pointwise transformer	8
PyTerrier [64]	Lexical	Traditional baselines	20

with 3 billion parameters. Additionally, we include 20 lexical retrieval models, e.g., BM25, PL2, etc., from PyTerrier, and for the duoT5 plugin of PyTerrier, we obtain 3 variants by using different duoT5 models, again, including the SOTA with 3 billion parameters. For all retrieval approaches, we keep all parameters at their defaults. ChatNoir makes heavy use of the different fields of the ClueWeb documents during retrieval and serves only as full-rank software, while all other pieces of software use the "default text" that we implemented in `ir_datasets`. The lexical approaches in PyTerrier and the dense approaches in BEIR can work as full-rank or re-rank software (we do not count "duplicates", the full-rank variants are always realized from a first component building the index and a second component retrieving from the index). All duoT5 and PyGaggle variants only work as re-rankers. For ColBERT, we only add the re-rank variant, as ColBERT indices become very large.

We execute all 50 retrieval models on all 32 tasks in TIRA. To increase the comparability of the results, we run each software as re-rankers, where the 50 retrieval approaches re-rank the ChatNoir ranking for the ClueWebs, and the full-rank results of PyTerrier BM25. We executed the Lexical approaches using 1 CPU and 10 GB of RAM, and all other models had additional access to 1 GeForce GTX 1080 GPU with 8 GB. Some models failed on this GPU because 8 GB was insufficient (ColBERT and 2 SBERT models failed on a few tasks, monoT5 and duoT5 with 3 billion parameters failed on all tasks). To handle those cases, we temporarily connected two additional runners with access to an A100 GPU with 40 GB to TIRA. All previously failed models were successful on the A100 GPU. TIRA manages metadata about the resources used to produce a run, making differences transparent.

Table 5 shows the aggregated results of running and evaluating all 50 retrieval models on all 31 tasks (we leave out the results on the ClueWeb22 as the Touché 2023 task is still ongoing). We report the effectiveness as nDCG@10, reporting the macro average in case a corpus is associated with multiple tasks. We report the scores of BM25, ColBERT, TAS-B, all three duoT5 variants, and monoT5 (in its default configuration with its default model). Since we have overall 20 lexical, 17 bi-encoder, and 8 PyGaggle approaches, we show for each of those groups the best, median, and worst score. All deep learning models were trained on MS MARCO, while lexical models ran at their default parameters. Consequently, all deep learning models substantially outperform lexical models on MS MARCO. However, this is not necessarily true on other corpora, where the deep learning models act in a zero-shot style. Lexical retrieval models achieve the highest effectiveness on three of the tasks (Args.me, ClueWeb09, and MEDLINE). Our results further show that switching the lexical baseline can have a substantial impact, as BM25 is not always the best choice. For instance, the best lexical model achieves on Args.me an nDCG@10 of 0.57 while BM25 achieves 0.43. The effectiveness gap between the best and the worst model of a group can be substantial, even for lexical models (e.g., the best model on Args.me achieves an nDCG@10 of 0.57 while the worst achieves 0.14). The model choice is negligible on other corpora, e.g., for lexical models on NFCorpus. The leaderboards aggregated in Table 5 enable the selection of competitive baselines for many tasks that were impossible before.

4.2 Case Study: Reproducibility Analysis

As an example of a post-hoc analysis enabled by TIREx, we analyze to which degree parts of the system ranking can be reproduced on different tasks with `repro_eval`. On TREC Deep Learning 2019 with nDCG@10 as measure, we observe all system preferences between all pairs of the 50 retrieval models. For each system preference on the TREC Deep Learning track 2019 (e.g., monoT5 with 0.71 being more effective than BM25 with 0.48 is a system preference observed on TREC DL 2019), we analyze their reproducibility on the other tasks with `repro_eval`.

For a given system preference, we use the system with the lower nDCG@10 score on TREC Deep Learning 2019 as the baseline, and the system with the higher score as the advanced system. With `repro_eval`, we study the reproduction of those system preferences on a different task for two dimensions [14]: (1) the effect ratio of the reproduction, and (2) the delta of the relative improvement of the reproduction. The effect ratio measures to which degree the effect of the improvement of the advanced run against the baseline run can be reproduced on the new task (1 indicates a perfect reproduction, values between 0 and 1 indicate reproductions with diminished improvements on the new task, and 0 indicates failed reproductions). The delta relative improvement compares the relative effect of the improvement of the advanced run against the baseline run (0 indicates perfect reproductions, values below 0 indicate an increased relative improvement of the advanced run on the new task, values between 0 and below 1 indicate a smaller relative improvement, and 1 indicates a failed reproduction).

Table 5: Effectiveness scores (nDCG@10) on 14 corpora (31 tasks; ClueWeb22B excluded as no judgments yet) for selected approaches and the best, median, and worst of each group (scores macro-averaged for corpora with multiple associated tasks).

Corpus	ChatNoir	Lexical				Late Int.	Bi-Encoder				duoT5			PyGaggle			
		BM25	Best	Median	Worst	ColBERT	TAS-B	Best	Median	Worst	Base	Large	3b	MonoT5	Best	Median	Worst
Antique	—	0.51	0.53	0.51	0.36	0.47	0.40	0.49	0.44	0.30	0.54	0.46	0.52	0.51	**0.54**	0.51	0.45
Args.me	—	0.43	**0.57**	0.43	0.14	0.26	0.17	0.33	0.24	0.13	0.33	0.29	0.29	0.30	0.39	0.34	0.27
CORD-19	—	0.28	0.64	0.55	0.21	0.58	0.50	**0.70**	0.60	0.50	0.66	0.61	0.66	0.69	0.69	0.63	0.55
ClueWeb09	0.16	0.18	**0.24**	0.18	0.12	0.17	0.16	0.20	0.17	0.13	0.15	0.15	0.18	0.17	0.19	0.17	0.12
ClueWeb12	**0.36**	0.24	0.27	0.25	0.14	0.23	0.25	0.28	0.26	0.23	0.33	0.30	0.35	0.26	0.28	0.26	0.23
Cranfield	—	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Disks4+5	—	0.44	0.46	0.44	0.37	0.46	0.39	0.49	0.43	0.37	0.45	0.38	0.44	0.53	**0.57**	0.53	0.43
GOV	—	0.22	0.24	0.22	0.15	0.23	0.22	0.27	0.24	0.21	0.19	0.15	0.22	0.26	**0.29**	0.26	0.22
GOV2	—	0.47	0.49	0.44	0.25	0.45	0.34	0.46	0.42	0.34	0.47	0.43	0.48	0.48	**0.51**	0.48	0.41
MS MARCO	—	0.49	0.50	0.48	0.37	0.69	0.64	0.71	0.66	0.64	0.64	0.57	0.63	0.71	**0.74**	0.71	0.63
MEDLINE	—	0.34	**0.42**	0.27	0.18	0.25	0.14	0.26	0.21	0.14	0.34	0.32	0.36	0.25	0.35	0.27	0.24
NFCorpus	—	0.27	0.28	0.27	0.26	0.27	0.25	0.29	0.26	0.24	0.28	0.24	0.29	0.30	**0.31**	0.30	0.28
Vaswani	—	0.45	0.46	0.45	0.30	0.43	0.34	0.44	0.38	0.22	0.41	0.34	0.46	0.31	**0.48**	0.41	0.08
WaPo	—	0.38	0.39	0.37	0.24	0.43	0.34	0.43	0.37	0.33	0.40	0.28	0.40	0.45	**0.49**	0.45	0.40
Avg.	—	0.34	0.39	0.35	0.22	0.35	0.30	0.38	0.33	0.27	0.37	0.32	0.38	0.37	**0.42**	0.38	0.31

Table 6: Reproducibility of system preferences from TREC DL 2019 on selected tasks. We report the success rate in percent (effect ratio > 0) and the 25 %, 50 %, and 75 % quantiles for the effect ratio and delta relative improvement.

Benchmark	Rank	Succ.	Effect Ratio			Delta Rel. Impr.		
			25 %	50 %	75 %	25 %	50 %	75 %
TREC DL 2020	1	88.1	0.68	0.90	1.11	-0.03	0.02	0.08
Touché 2020 (Task 2)	2	77.1	0.12	0.38	0.73	-0.09	0.04	0.17
Web Track 2004	3	75.5	0.01	0.29	0.89	-0.07	0.10	0.31
TREC-7	4	73.9	-0.03	0.31	1.11	-0.02	0.12	0.34
Core 2018	5	70.2	-0.05	0.24	0.90	-0.03	0.13	0.35
NFCorpus	10	66.4	-0.06	0.06	0.32	0.02	0.23	0.42
Web track 2003	15	57.8	-0.14	0.04	0.23	-0.08	0.15	0.36
Web track 2009	20	44.1	-0.40	-0.04	0.26	0.00	0.30	0.52
Web track 2010	25	36.3	-0.49	-0.14	0.18	0.03	0.32	0.59
Web track 2013	30	31.0	-0.43	-0.21	0.13	0.06	0.30	0.63

Table 6 shows the results of our reproducibility analysis with `repro_eval`. We report the ratio of system preferences with a successful reproduction of the effect ratio and the 25 %, 50 %, and 75 % quantile for the effect ratio and the relative delta improvement. We sort the tasks by the percentage of successful reproductions, showing the first five tasks with the highest success rate, and then continue with a step size of 5. Not surprisingly, the reproductions to the TREC Deep Learning Track of 2020 are excellent: 88.1 % of preferences have an effect ratio above 0. This ratio declines fast, as the task on rank 15 had only a success rate of 57.8 %. Analyzing the results for the quantiles yields similar observations (e.g., 50 % of the system preferences have an almost perfect effect ratio of 0.90 or higher for TREC DL 2020 while this declines fast, as already the task on rank 15 as a median effect ration of 0.04). Hence, our reproduction analysis indicates that out of the many tasks available in TIREx, a smaller subset might suffice for prototyping, as tasks that reproduce previously analyzed tasks might be skipped.

4.3 Predicted Impact of the Platform

TIREx can have a substantial conceptual impact as there might be no alternative to blinded evaluations in the future (given the practice of training LLMs on basically all available tasks [19]). Additionally, the platform eases the organization of IR experiments. Shared task organizers can simply provide the well-documented open-source baselines from TIRA as starting points for the participants and can also use them to ensure some variety in the pooling process, especially for tasks that attract few participants. For shared tasks that run multiple years, organizers can automatically re-run all approaches submitted to previous editions, allowing for a transparent tracking of progress. The platform combines leaderboards with immutable, dockerized software, enabling researchers and reviewers to use and execute good baselines in a few lines of code.

The submission platform TIRA proved robust after its complete redevelopment [41]: two large-scale NLP tasks with software submissions used TIRA at SemEval 2023 (171 registered teams, 71 teams submitted results, 647 runs produced by software submissions). Our initial experiments with TIREx produced 1,600 runs, showing the platform to be robust and to have the potential of changing how we conduct IR experiments in the future.

5 CONCLUSION

With TIREx—the IR Experiment Platform—we aim to substantially ease conducting (blinded) IR experiments and organizing "always-on" reproducible shared tasks with software submissions. Our new platform integrates `ir_datasets`, `ir_measures`, and PyTerrier with TIRA. Retrieval workflows can be executed on-demand via cloud-native orchestration, reducing the efforts for reproducing IR experiments as all approaches submitted to TIREx as software can be re-executed in post-hoc experiments. The platform has no lock-in effect, as archived experiments are fully self-contained and work stand-alone. With keeping test data private, TIREx also addresses a potential further "professionalization" of IR experiments similar to fields like medicine, where blinded experiments are the norm. TIREx is open to the IR community and ready to incorporate more collections and retrieval approaches.

ACKNOWLEDGMENTS

This work has been partially supported by the OpenWebSearch.eu project (funded by the EU; GA 101070014).

REFERENCES

[1] J. Arguello, F. Diaz, J. Lin, and A. Trotman. SIGIR 2015 workshop on reproducibility, inexplicability, and generalizability of results (RIGOR). *SIGIR 2015*. 1147–1148.

[2] T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel. EvaluatIR: An online tool for evaluating and comparing IR systems. *SIGIR 2009*. 833.

[3] T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel. Improvements that don't add up: Ad-hoc retrieval results since 1998. *CIKM 2009*. 601–610.

[4] P. Bailey, A. Moffat, F. Scholer, and P. Thomas. UQV100: A test collection with query variability. *SIGIR 2016*. 725–728.

[5] M. Baker. 1,500 scientists lift the lid on reproducibility. *Nature* 533, 7604 (2016).

[6] R. Benham, L. Gallagher, J. M. Mackenzie, B. Liu, X. Lu, F. Scholer, J. Shane Culpepper, and A. Moffat. RMIT at the 2018 TREC CORE Track. *TREC 2018*.

[7] J. Bevendorff, B. Stein, M. Hagen, and M. Potthast. Elastic ChatNoir: Search engine for the ClueWeb and the Common Crawl. *ECIR 2018*. 820–824.

[8] A. Bondarenko, M. Fröbe, J. Kiesel, F. Schlatt, V. Barriere, B. Ravenet, L. Hemamou, S. Luck, J. Heinrich Reimer, B. Stein, M. Potthast, and M. Hagen. Overview of Touché 2023: Argument and causal retrieval. *ECIR 2023*. 527–535.

[9] A. Bondarenko, M. Fröbe, J. Kiesel, S. Syed, T. Gurcke, M. Beloucif, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, and M. Hagen. Overview of Touché 2022: Argument retrieval. *CLEF 2022*. 311–336.

[10] A. Bondarenko, L. Gienapp, M. Fröbe, M. Beloucif, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, and M. Hagen. Overview of Touché 2021: Argument retrieval. *CLEF 2021*. 450–467.

[11] L. Bonifacio, H. Abonizio, M. Fadaee, and R. Nogueira. InPars: Unsupervised dataset generation for information retrieval. *SIGIR 2022*. 2387–2392.

[12] V. Boteva, D. Gholipour Ghalandari, A. Sokolov, and S. Riezler. A full-text learning to rank dataset for medical information retrieval. *ECIR 2016*. 716–722.

[13] L. Boytsov and E. Nyberg. Flexible retrieval with NMSLIB and FlexNeuART. *NLP-OSS 2020*. 32–43.

[14] T. Breuer, N. Ferro, N. Fuhr, M. Maistro, T. Sakai, P. Schaer, and I. Soboroff. How to measure the reproducibility of system-oriented IR experiments. *SIGIR 2020*. 349–358.

[15] T. Breuer, N. Ferro, M. Maistro, and P. Schaer. repro_eval: A Python interface to reproducibility measures of system-oriented IR experiments. *ECIR 2021*. 481–486.

[16] T. Breuer, J. Keller, and P. Schaer. ir_metadata: An extensible metadata schema for IR experiments. *SIGIR 2022*. 3078–3089.

[17] T. Breuer, P. Schaer, N. Tavakolpoursaleh, J. Schaible, B. Wolff, and B. Müller. STELLA: Towards a framework for the reproducibility of online search experiments. *OSIRRC at SIGIR 2019*. 8–11.

[18] S. Büttcher, C. L. A. Clarke, and I. Soboroff. The TREC 2006 Terabyte track. *TREC 2006*.

[19] H. Won Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. Shane Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Y. Zhao, Y. Huang, A. M. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei. Scaling instruction-finetuned language models. arXiv:2210.11416 (2022).

[20] R. Clancy, N. Ferro, C. Hauff, J. Lin, T. Sakai, and Z. Z. Wu. Overview of the 2019 Open-Source IR Replicability Challenge (OSIRRC 2019). *OSIRRC at SIGIR 2019*. 1–7.

[21] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 Terabyte track. *TREC 2004*.

[22] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 Web track. *TREC 2009*.

[23] C. L. A. Clarke, N. Craswell, I. Soboroff, and G. V. Cormack. Overview of the TREC 2010 Web track. *TREC 2010*.

[24] C. L. A. Clarke, N. Craswell, I. Soboroff, and E. M. Voorhees. Overview of the TREC 2011 Web track. *TREC 2011*.

[25] C. L. A. Clarke, N. Craswell, and E. M. Voorhees. Overview of the TREC 2012 Web track. *TREC 2012*.

[26] C. L. A. Clarke, F. Scholer, and I. Soboroff. The TREC 2005 Terabyte track. *TREC 2005*.

[27] C. Cleverdon. The Cranfield tests on index language devices. *ASLIB Proceedings*, 1967, 173–192.

[28] C. Cleverdon. The significance of the Cranfield tests on index languages. *SIGIR 1991*. 3–12.

[29] K. Collins-Thompson, P. N. Bennett, F. Diaz, C. Clarke, and E. M. Voorhees. TREC 2013 Web track overview. *TREC 2013*.

[30] K. Collins-Thompson, C. Macdonald, P. N. Bennett, F. Diaz, and E. M. Voorhees. TREC 2014 Web track overview. *TREC 2014*.

[31] C. Costello, E. Yang, D. Lawrie, and J. Mayfield. Patapasco: A Python framework for cross-language information retrieval experiments. *ECIR 2022*.

[32] N. Craswell and D. Hawking. Overview of the TREC-2002 Web track. *TREC 2002*.

[33] N. Craswell and D. Hawking. Overview of the TREC 2004 Web track. *TREC 2004*.

[34] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu. Overview of the TREC 2003 Web track. *TREC 2003*. 78–92.

[35] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos. Overview of the TREC 2020 Deep Learning track. *TREC 2020*.

[36] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. Overview of the TREC 2019 Deep Learning track. *TREC 2019*.

[37] N. Ferro, N. Fuhr, K. Järvelin, N. Kando, M. Lippold, and J. Zobel. Increasing reproducibility in IR: Findings from the Dagstuhl seminar on "Reproducibility of Data-Oriented Experiments in E-Science". *SIGIR Forum* 50, 1 (2016), 68–82.

[38] N. Ferro, N. Fuhr, M. Maistro, T. Sakai, and I. Soboroff. Overview of CENTRE@CLEF 2019: Sequel in the systematic reproducibility realm. *CLEF 2019*. 287–300.

[39] N. Ferro, M. Maistro, T. Sakai, and I. Soboroff. Overview of CENTRE@CLEF 2018: A first tale in the systematic reproducibility realm. *CLEF 2018*.

[40] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. SPLADE v2: Sparse lexical and expansion model for information retrieval. arXiv:2109.10086 (2021).

[41] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, and M. Potthast. Continuous integration for reproducible shared tasks with TIRA.io. *ECIR 2023*. 236–241.

[42] N. Fuhr. Some common mistakes in IR evaluation, and how they can be avoided. *SIGIR Forum* 51, 3 (2017), 32–41.

[43] N. Fuhr. Proof by experimentation? Towards better IR research. *SIGIR Forum* 54, 2 (2020), 2:1–2:4.

[44] N. Fuhr. Proof by experimentation? Towards better IR research. *SIGIR 2020*. 2.

[45] L. Gao, X. Ma, J. Lin, and J. Callan. Precise zero-shot dense retrieval without relevance labels. arXiv:2212.10496 (2022).

[46] T. Gollub, B. Stein, S. Burrows, and D. Hoppe. TIRA: Configuring, executing, and disseminating information retrieval experiments. *TIR 2012 at DEXA*. 151–155.

[47] H. Hashemi, M. Aliannejadi, H. Zamani, and W. Bruce Croft. ANTIQUE: A non-factoid question answering benchmark. *ECIR 2020*. 166–173.

[48] W. R. Hersh, R. Teja Bhupatiraju, L. Ross, A. M. Cohen, D. Kraemer, and P. Johnson. TREC 2004 Genomics track overview. *TREC 2004*.

[49] W. R. Hersh, A. M. Cohen, J. Yang, R. Teja Bhupatiraju, P. M. Roberts, and M. A. Hearst. TREC 2005 Genomics track overview. *TREC 2005*.

[50] S. Hofstätter, S. Lin, J. Yang, J. Lin, and A. Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. *SIGIR 2021*. 113–122.

[51] F. Hopfgartner, T. Brodt, J. Seiler, B. Kille, A. Lommatzsch, M. A. Larson, R. Turrin, and A. Serény. Benchmarking news recommendations: The CLEF NewsREEL use case. *SIGIR Forum* 49, 2 (2015), 129–136.

[52] F. Hopfgartner, A. Hanbury, H. Müller, I. Eggel, K. Balog, T. Brodt, Gordon V. Cormack, J. Lin, J. Kalpathy-Cramer, N. Kando, Makoto P. Kato, A. Krithara, T. Gollub, M. Potthast, E. Viegas, and S. Mercer. Evaluation-as-a-service for the computational sciences: Overview and outlook. *Journal of Data and Information Quality* 10, 4 (2018), 15:1–15:32.

[53] R. Jagerman, K. Balog, and M. de Rijke. OpenSearch: Lessons learned from an online evaluation campaign. *Journal of Data and Information Quality* 10, 3 (2018), 13:1–13:15.

[54] V. Karpukhin, B. Oguz, S. Min, P. S. H. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih. Dense passage retrieval for open-domain question answering. *EMNLP 2020*. 6769–6781.

[55] O. Khattab and M. Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. *SIGIR 2020*. 39–48.

[56] J. Leipzig, D. Nüst, C. Tapley Hoyt, K. Ram, and J. Greenberg. The role of metadata in reproducible computational research. *Patterns* 2, 9 (2021), 100322.

[57] J. Lin. The neural hype and comparisons against weak baselines. *SIGIR Forum* 52, 2 (2018), 40–51.

[58] J. Lin, D. Campos, N. Craswell, B. Mitra, and E. Yilmaz. Fostering coopetition while plugging leaks: The design and implementation of the MS MARCO leaderboards. *SIGIR 2022*. 2939–2948.

[59] J. Lin, X. Ma, S. Lin, J. Yang, R. Pradeep, and R. Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. *SIGIR 2021*. 2356–2362.

[60] J. Lin and Q. Zhang. Reproducibility is a process, not an achievement: The replicability of IR reproducibility experiments. *ECIR 2020*. 43–49.

[61] S. MacAvaney, C. Macdonald, and I. Ounis. Streamlining evaluation with ir-measures. *ECIR 2022*. 305–310.

[62] S. MacAvaney, A. Yates, S. Feldman, D. Downey, A. Cohan, and N. Goharian. OpenNIR: A complete neural ad-hoc ranking pipeline. *WSDM 2020*. 845–848.

[63] S. MacAvaney, A. Yates, S. Feldman, D. Downey, A. Cohan, and N. Goharian. Simplified data wrangling with ir_datasets. *SIGIR 2021*. 2429–2436.

[64] C. Macdonald, N. Tonellotto, S. MacAvaney, and I. Ounis. PyTerrier: Declarative experimentation in Python from BM25 to dense retrieval. *CIKM 2021*. 4526–4533.

[65] A. Mallia, M. Siedlaczek, J. M. Mackenzie, and T. Suel. PISA: Performant indexes and search for academia. *OSIRRC at SIGIR 2019*. 50–56.

[66] A. Moffat. Batch evaluation metrics in information retrieval: Measures, scales, and meaning. *IEEE Access* 10 (2022), 105564–105577.

[67] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A human cenerated machine reading comprehension dataset. *CoCo at NIPS 2016*.

[68] R. Frassetto Nogueira, W. Yang, K. Cho, and J. Lin. Multi-stage document ranking with BERT. arXiv:1910.14424 (2019).

[69] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson. Terrier information retrieval platform. *ECIR 2005*. 517–519.

[70] B. Piwowarski. Experimaestro and Datamaestro: Experiment and dataset managers (for IR). *SIGIR 2020*. 2173–2176.

[71] M. Potthast, T. Gollub, M. Wiegmann, and B. Stein. TIRA integrated research architecture. *Information Retrieval Evaluation in a Changing World*, 2019. 123–160.

[72] R. Pradeep, R. Nogueira, and J. Lin. The Expando-Mono-Duo design pattern for text ranking with pretrained sequence-to-sequence models. arXiv:2101.05667 (2021).

[73] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *EMNLP-IJCNLP 2019*. 3980–3990.

[74] K. Roberts, D. Demner-Fushman, E. M. Voorhees, W. R. Hersh, S. Bedrick, and A. J. Lazar. Overview of the TREC 2018 Precision Medicine track. *TREC 2018*.

[75] K. Roberts, D. Demner-Fushman, E. M. Voorhees, W. R. Hersh, S. Bedrick, A. J. Lazar, and S. Pant. Overview of the TREC 2017 Precision Medicine track. *TREC 2017*.

[76] T. Sakai. On Fuhr's guideline for IR evaluation. *SIGIR Forum* 54, 1 (2020), 12:1–12:8.

[77] T. Sakai, N. Ferro, I. Soboroff, Z. Zeng, P. Xiao, and M. Maistro. Overview of the NTCIR-14 CENTRE task. *NTCIR 2019*.

[78] T. Sakai, S. Tao, Z. Zeng, Y. Zheng, J. Mao, Z. Chu, Y. Liu, M. Maistro, Z. Dou, N. Ferro, et al. Overview of the NTCIR-15 We Want Web with CENTRE (WWW-3) task. *NTCIR 2020*.

[79] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. *NeurIPS Datasets and Benchmarks 2021*.

[80] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, Y. Almirantis, J. Pavlopoulos, N. Baskiotis, P. Gallinari, T. Artières, A. Ngonga Ngomo, N. Heino, É. Gaussier, L. Barrio-Alvers, M. Schroeder, I. Androutsopoulos, and G. Paliouras. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics* 16 (2015), 138:1–138:28.

[81] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explor.* 15, 2 (2013), 49–60.

[82] E. Voorhees. Overview of the TREC 2004 Robust Retrieval track. *TREC 2004*.

[83] E. M. Voorhees. NIST TREC Disks 4 and 5: Retrieval test collections document set. 1996.

[84] E. M. Voorhees. The philosophy of information retrieval evaluation. *CLEF 2001*. 355–370.

[85] E. M. Voorhees. The evolution of Cranfield. *Information Retrieval Evaluation in a Changing World*, 2019. 45–69.

[86] E. M. Voorhees, T. Alam, S. Bedrick, D. Demner-Fushman, W. R. Hersh, K. Lo, K. Roberts, I. Soboroff, and L. Lu Wang. TREC-COVID: Constructing a pandemic information retrieval test collection. *SIGIR Forum* 54, 1 (2020), 1:1–1:12.

[87] E. M. Voorhees, N. Craswell, and J. Lin. Too many relevants: Whither Cranfield test collections? *SIGIR 2022*. 2970–2980.

[88] E. M. Voorhees and D. Harman. Overview of the seventh text retrieval conference (TREC-7). *TREC 1998*.

[89] E. M. Voorhees and D. Harman. Overview of the eighth text retrieval conference (TREC-8). *TREC 1999*.

[90] E. M. Voorhees, I. Soboroff, and J. Lin. Can old TREC collections reliably evaluate modern neural retrieval models? arXiv:2201.11086 (2022).

[91] L. Lu Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. Kinney, Z. Liu, W. Merrill, P. Mooney, D. A. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. Raymond, D. S. Weld, O. Etzioni, and S. Kohlmeier. CORD-19: The Covid-19 open research dataset. arXiv:2004.10706 (2020).

[92] L. Xiong, C. Xiong, Y. Li, K. Tang, J. Liu, P. N. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *ICLR 2021*.

[93] D. Yadav, R. Jain, H. Agrawal, P. Chattopadhyay, T. Singh, A. Jain, S. Singh, S. Lee, and D. Batra. EvalAI: Towards better evaluation systems for AI agents. arXiv:1902.03570 (2019).

[94] P. Yang, H. Fang, and J. Lin. Anserini: Enabling the use of Lucene for information retrieval research. *SIGIR 2017*. 1253–1256.

[95] A. Yates, S. Arora, X. Zhang, W. Yang, K. Martin Jose, and J. Lin. Capreolus: A toolkit for end-to-end neural ad hoc retrieval. *WSDM 2020*. 861–864.

[96] X. Zhang, N. Thakur, O. Ogundepo, E. Kamalloo, D. Alfonso-Hermelo, X. Li, Q. Liu, M. Rezagholizadeh, and J. Lin. Making a MIRACL: Multilingual information retrieval across a continuum of languages. arXiv:2210.09984 (2022).

[97] J. Zobel. When measurement misleads: The limits of batch assessment of retrieval systems. *SIGIR Forum* 56, 1 (2022), 12:1–12:20.