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 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 or 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. Using Docker images of 50 standard retrieval approaches, we automatically evaluated all approaches on all tasks ($50 \cdot 32 = 1{,}600$ runs) in less than a week on a midsize cluster (1,620 CPU cores and 24 GPUs). This instance of TIREx is open for submissions and will be integrated with the IR Anthology, as well as released open source.

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 [28], which became the de facto standard for shared tasks hosted at TREC [92] and beyond. 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 [91]. Finally, the 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 laboratories working independently on the same task, the community draws on the "wisdom of crowds" while ensuring 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, 65]. Second, run submissions require that participants have access to the test topics, which has severe implications [45], 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 [73] to the Flan-T5 prompting model [20] 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

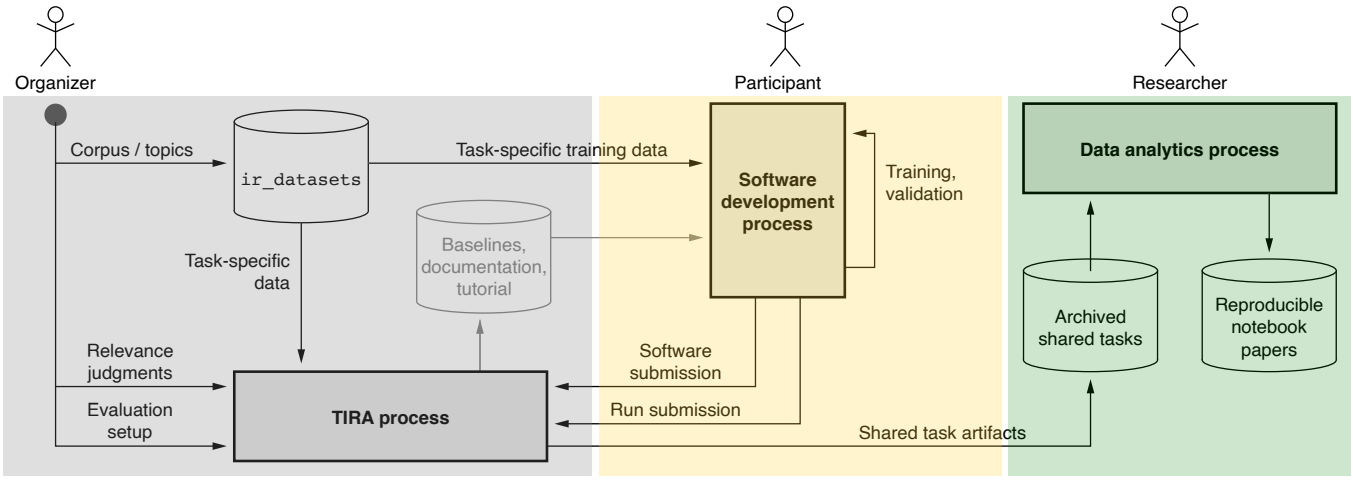

Figure 1: Overview of typical shared task-like IR experiments and how the tools in TIREx support them.

of tools for working with IR data (`ir_datasets` [68]), for executing retrieval pipelines (PyTerrier [69]), and for evaluating IR systems (`ir_measures` [66]) with the TIRA Integrated Research Architecture [43], a continuous integration service for reproducible shared tasks and experiments. TIREx is designed to for reproducibility through software submissions while keeping an experimenter's or task organizer's workload 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. A substantial efficiency boost comes from integrating GPU cores and result caching into the platform to accelerate neural IR approaches. As a proof of concept, we conducted a large-scale evaluation of 50 "standard" retrieval approaches on 32 shared retrieval tasks (based on 15 corpora with a total of 1.9 billion documents). This experiment consists of 1,600 runs and was started by just clicking a button. It finished unattended in less than a week.

2 BACKGROUND AND RELATED WORK

We review ad hoc retrieval experiments in evaluation campaigns, common problems and pitfalls in IR experiments, best practices for leaderboards, existing reproducibility initiatives, and tools to support reproducibility. Insights from all these areas 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 [92]. In the 1960s, the Cranfield experiments [28, 29] 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 [92]. 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 [92]. 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 [92]. Although this completeness assumption is reasonable for tasks with a diverse set of submitted runs pooled at high depth [97], recent observations suggest that scenarios with many relevant documents per query (e.g., corpora with many duplicates [94]) or with topics representing broad information needs [86] 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 different baseline retrieval systems is directly available and can be applied to any imported retrieval task.

Common Problems and Pitfalls in IR Experiments. Even though the current discussion about how to conduct IR experiments [44, 83, 104] includes some controversial points (e.g., whether MRR should be abandoned [44] or not [72, 83]), there is still a 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) [46]. Still, external validity of IR experiments remains an open problem [45]. 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. [86] 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., [12, 41, 47])—often even without clearly justifying the selection. In contrast, a software in TIREx can rather easily be evaluated against many if not 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 or "wrong" baselines (i.e., not the best or most reasonable previous approaches). Any improvements

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

claimed in such studies are not really meaningful [3, 62]. 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 [62]. 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 [62]. 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) 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" [62].

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 [103] received 25 submissions within one week, and the MS MARCO leaderboard [63] 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 [63]. Lin et al. [63] propose best practices, inspired by previous problems of the Netflix prize.[5] Most importantly, Lin et al. note that, while submissions to a 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. [63].

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, 21] or CENTRE [39, 40, 84, 85]. 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 [65] 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 [75] were fully reproducible out of the box, while two 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 (i.e., the operating system with its packages). To mitigate the problem of changing platform dependencies, the follow-up iteration of OSIRRC [21] 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 in 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 minimize the TIREx requirements: 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 [18, 43, 54, 56, 87, 88, 100]. For instance, as documenting the metadata of experiments improves reproducibility [61], `ir_metadata` [17] simplifies the documentation of IR experiments according to the PRIMAD model [38] (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 [55]. 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 supports blinded experimentation and as it is based on (private) git repositories hosted on GitLab or 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. We substantially extend 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 have constructed the Information Retrieval Experiment Platform (TIREx) to facilitate reproducible, shared task-style IR experiments based on software submissions. This has been achieved by integrating `ir_datasets`, `ir_measures`, and PyTerrier into TIRA. We anticipate the sustained availability and maintenance of these components, as evidenced by TIRA's and PyTerrier's consistent upkeep since 2012 [48] and 2020 [70], respectively, and the growing popularity of `ir_datasets` in recent years. Previously, conducting shared task-style IR experiments within TIRA was already possible, but required significant effort from both organizers and participants due to their unique nature, compared to standard Machine Learning or Natural Language Processing experiments. IR experiments typically involve intermediate artifacts (like indexes), and retrieval systems

[5] 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] codalab.org, eval.ai, stella-project.org, tira.io

involve multi-stage "telescoping" pipelines.[8] To address these requirements, TIREx extends TIRA with common IR tools for data access, indexing, retrieval, and evaluation, and implements multi-stage pipelines on top of TIRA's underlying execution protocol. Below, we elaborate on how TIREx supports IR experiments, discuss the interaction between integrated tools, provide examples of using available retrieval approaches in TIREx, and demonstrate how TIREx promotes post-experiment replicability and reproducibility through declarative PyTerrier pipelines.

3.1 Experiments in the IR Experiment Platform

As illustrated in Figure 1, TIREx facilitates the entire process of conducting retrieval experiments. It allows shared task organizers and individual experimenters to import data and utilize any pre-existing retrieval software submitted to TIREx as baselines. Following that, submissions of new retrieval approaches for evaluation can be made as software submissions or, if enabled, also as run submissions. Any submission can be accompanied by descriptive annotations and metadata; for instance, run submissions can be grouped to denote that they were generated by the same retrieval approach for multiple retrieval tasks. By providing relevance judgments, organizers or experimenters can directly evaluate all available runs.

To incorporate a new corpus and topics into TIREx, they can be easily added to `ir_datasets`, utilizing a private branch if the data is sensitive. This data can then be imported by TIRA through a Docker image with a matching `ir_datasets` installation. Participants submit their software as Docker images as well. TIRA ensures their reproducibility and prevents test data leaks by executing them in a sandbox. Among other things, the sandbox disables Internet connectivity for the running software, which ensures that the software and its dependencies are fully installed and no data is sent to unauthorized third parties. Participants can provide additional data their software needs during execution by uploading it to TIRA. This is particularly useful for non-reproducible elements of a submission, such as manual query reformulations. TIREx also provides a "starter implementation" for five commonly used IR research frameworks, which participants can use as a development base. The simplest starter uses BM25 retrieval, which is implemented using a few lines of declarative PyTerrier code in a Jupyter notebook.[9]

TIREx allows for software submissions to be executed on demand within a cloud-based execution environment, utilizing GitLab or GitHub CI/CD pipelines. In order to meet varying demand, experiment organizers can incorporate additional runners as necessary. TIREx maintains a comprehensive record of every artifact of a retrieval experiment within a specific git repository (Figure 1, right), which can be exported and published. This "archived shared task" is entirely self-contained, enabling the independent re-execution of approaches with identical or differing data using PyTerrier pipelines. The availability of every software that generated a run as part of the repository makes it a key outcome and asset of an experiment. Consequently, TIREx facilitates "always-on" shared tasks for the IR community, along with an extensive variety of ablation studies.

[8]For instance, the Mono-Duo-Reranking pipelines [79], where a more complex re-ranker improves part of the ranking of a less complex one ahead in the pipeline.
[9]github.com/tira-io/ir-experiment-platform#starter-for-pyterrier-in-jupyter

3.2 Reproducible Shared Tasks with TIRA

TIRA is used to handle software submissions in shared tasks since 2012 [48, 77]—the CLEF labs PAN and Touché being two long-running examples.[10] A first version of TIRA provided participants with access to virtual machines to deploy their software. However, this setup required manual overhead on the part of organizers thus did not scale far beyond these two events. Moreover, software reexecution was possible in principle and has been demonstrated once at scale [49], but proved to be error-prone and required manual bug fixing inside the virtual machines as participant software was not robust against slight data format variations that were in principle supported by the underlying formatting schema. This also has prevented external researchers from reproducing the collected software for a given task at scale.

Meanwhile, Docker has gained maturity and widespread adoption and is now supported by many cluster computing frameworks such as Kubernetes. Especially their integration as GitHub and GitLab runners made automatic deployment widely available. Hence, TIRA was completely redeveloped based on the now industry-standard CI/CD pipelines (continuous integration and deployment) using Git, Docker, and Kubernetes [43]. In the new version of TIRA, participants upload their software implemented in Docker images to a private Docker registry dedicated to their team, ensuring that different teams do not influence each other while a shared task is running—the approaches can remain private until the task ends. For on-demand execution, TIRA presently runs the software on our Kubernetes cluster (1,620 CPU cores, 25.4 TB RAM, 24 GeForce GTX 1080 GPUs). This version of TIRA was first used in two NLP tasks hosted at SemEval 2023 to which 71 of 170 registered teams submitted 647 runs based on software submissions [42, 60].

While preparing the TIRA setup for the retrieval-oriented Touché 2023 tasks [9], we realized that the new TIRA still had some shortcomings. There was no unified access to IR data, no separation between full-rank or re-rank approaches, no modularization of software components with caching, and typical IR workflows were only realizable inefficiently or via workarounds. For instance, full-rank retrieval in TIRA would have required any software to build an index from scratch and different re-rank approaches would each have to re-create the baseline rankings. A re-ranking approach for the ClueWeb22-based Task 2 of Touché 2023 [9], for example, should have been able to use a ChatNoir baseline ranker [7] from within TIRA, but our pilot experiments showed that retrieving the top-1000 ChatNoir results for some set of 50 Touché topics [8–11] takes 54 to 134 minutes (ChatNoir requests can fail so that a client has to retry the requests). Blocking GPUs—often required by re-rankers—for such a long time would waste resources and the baseline's top-1000 results should ideally be cached so that different re-rankers can directly use them. To solve all these problems, we substantially expanded TIRA and redeveloped major parts to integrate `ir_datasets`, `ir_measures`, and PyTerrier.

3.3 Standardized Data Access with `ir_datasets`

The `ir_datasets` toolkit [68] provides a standard interface to access over 200 corpora and over 500 topic sets frequently used in IR experiments. The data is kept up-to-date (e.g., most TREC 2022

[10]pan.webis.de, touche.webis.de

tracks are included) and processing documents or topics is possible via a single line of Python code. Thus, `ir_datasets` already serves as a common data layer in numerous IR frameworks and tools (e.g., Capreolus [102], Experimaestro-IR [76], FlexNeuART [14], OpenNIR [67], Patapsco [32], PyTerrier [69]) and can be easily incorporated by most others (e.g., Anserini [101], PISA [71]). We integrate `ir_datasets` into TIRA via Docker images that can import complete corpora (for full-rank approaches) and that can create re-rankings for any given run file (for re-ranking approaches). To configure an IR experiment in TIRA, the experiment organizer only needs to provide an `ir_datasets` Docker image—standard images are available in TIREx but other images are also possible (e.g., for proprietary data). In the following, we further describe the new 'default_text' fields that we added to `ir_datasets` to enable re-using single-field retrieval software on corpora with multiple text fields, and we describe how the integration of `ir_datasets` into Docker images that run on-demand also ensures interchangeability and compatibility of retrieval components in retrieval pipelines.

Re-Usable Retrieval Software via default_text. While some corpora have a single text field for each document (e.g., the MS MARCO passage ranking corpus [36, 37, 73]), others provide rich structural information or metadata (e.g., the Touché corpora [10, 11] with structured arguments or comparison aspects). Similarly, some retrieval tasks have a single text field per topic (e.g., Antique [50]), while others provide metadata for each topic and/or multiple fields for versions of a query (e.g., TREC Precision Medicine [81, 82]).

Corpora and retrieval tasks with fine-grained structure usually address the development of built-for-purpose retrieval systems that exploit the task-specific setup. For instance, an argument retrieval system submitted to Touché may specifically focus on the argumentative premises contained in a document, and an approach in the Precision Medicine track may use a query's structure to adjust the relevance criteria. Instead, corpora and tasks with single fields for document texts and queries often rather address "general search" scenarios (i.e., retrieval approaches that can be applied in a variety of contexts rather than targeting one specific case). To also enable the evaluation of such general purpose retrieval systems (that expect a single document text field and a single query field) on data with more fields, we created default_text fields for every dataset in `ir_datasets`. There often is a natural choice for a document's or a query's "default text" (e.g., we simply concatenated the two fields 'title' and 'abstract' of MEDLINE documents as the default document text and we often selected a TREC topic's title as the query text—after a manual review). Still, there also are more difficult cases for which we then carefully tried to select the most important content of the documents or topics—being open to corrective pull requests from the community. The new default_text fields now are part of the `ir_datasets` package and thus also applicable in TIREx to ensure reusability of single-field retrieval approaches on data originally only available with multiple fields.

Ensuring Compatibility of Modularized Retrieval Stages. TIREx aims to support experiments in which components for the individual stages of modularized retrieval pipelines can be easily replaced and compared without having to adapt the complete retrieval software each time. Therefore, TIRA distinguishes between two types of retrieval approaches: (1) full-rank approaches with a document corpus and topics as input, and (2) re-rankers with a re-rank file as input (basically, query–document pairs). From any retrieval software's output, a re-rank file can be automatically created and cached in TIREx by the `ir_datasets` integration. As the structure of these re-rank files always is the same, any re-ranker can easily run on the output of any previous retrieval approach. Note that some data in `ir_datasets` can not be downloaded from the Web and/or requires license agreements (e.g., the ClueWeb and GOV corpora). As we have valid license agreements on our local TIREx instance, we can directly mount such data into the `ir_datasets` container, but, by default, then only show effectiveness scores for a run and no retrieval results (i.e., participants do not get access to the corpus as their software is executed in a sandbox and all outputs other than effectiveness scores are not shown on confidential datasets).

Table 1 shows the data fields that the `ir_datasets` integration makes available. For full-rank software, the `documents.jsonl.gz` file for each document contains an identifier 'docno', the new default_text in the field 'text', and all original structured fields of a document in 'original_document'. The `topics.jsonl.gz` file for each topic contains an identifier 'qid', the new default_text in the field 'query', and all original structured fields of a topic in 'original_topic'. For re-rankers, the `ir_datasets` integration creates a file `re-rank.jsonl.gz` from the output of a previous retrieval stage (i.e., the run file), where each entry contains query–document pairs to be reranked along their score and rank assigned by the previous stage. When relevance judgments exist, the `ir_datasets` integration can also make them available in a `qrels.txt` file so that the evaluator software specified by the experiment organizer can automatically evaluate submitted retrieval approaches.

3.4 Sanity-checked Evaluation with `ir_measures`

TIRA can automatically evaluate run files (created by retrieval software submissions or uploads) via an `ir_measures` evaluator. First, the evaluator performs a sanity check to test whether a run file can be parsed and warns of potential errors (e.g., score ties, NaN scores, empty result sets, unknown queries, scores contradicting the ranks, etc.). Then, if relevance judgments have been provided, the evaluator derives all specified measures averaged over all queries and per query (suitable for significance tests).

3.5 Reproducible IR Pipelines with TIRA

To improve the efficiency of common IR workflows in TIREx, we redeveloped and extended TIRA's ability to define and run modularized software even spanning multiple Docker images. All software in TIRA is immutable so that outputs of one software (e.g., an index) can be cached and reused by another software.

Modularized Software with Multiple Components. Retrieval software in TIRA can have multiple components that form a sequence similar to UNIX pipes or even a directed acyclic graph (DAG). Each component has a Docker image with a command to be executed and can have none, one, or many preceding components, respectively. TIRA passes the corresponding input and output directories to each component via three variables (cf. Table 2). The variable `$inputDataset` points to the directory that contains the actual input (e.g., `re-rank.jsonl.gz` for re-ranking software). The variable

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).

`$inputRun` is only available when a component has preceding components and then points to a directory with all outputs of the directly preceding components. The variable `$outputDir` specifies the location where TIRA expects a component's outputs. All three variables `$inputDataset`, `$inputRun`, and `$outputDir` can be included in the command to be executed but are also available as environment variables within a container.

Components for Additional Data. A retrieval approach might use data unavailable in an experiment's original corpus and topics. An example are user query variants for ClueWeb or Common Core topics [4, 6]—using the variants would count as a "manual" run at TREC. TIREx supports such cases of additional data via file uploads which becomes available to subsequent software via the `$inputRun` variable. Uploaded files can be grouped, documented, and configured as components treated like a software component (e.g., to precede some ranking component). Thus, TIREx supports manual runs and other kinds of use cases, but isolates such steps as much as possible to keep the software part of a pipeline replicable.

Defining Retrieval Pipelines. Figure 2 illustrates the conceptual data flow for the two simplest-possible sequence and DAG retrieval pipelines, respectively. The upper pipeline shows a full-rank approach that first creates an index (component 'Index Corpus' with output `file-01` and `file-02`) that TIRA then makes available as `$inputRun` for the second component 'BM25 Retrieval'. Figure 3 shows how a BM25 retrieval component that depends on a PyTerrier index can be defined in TIRA. Since many different components of a software may use a created artifact like an index, we cache all outputs to make pipelines more efficient.

The lower pipeline in Figure 2 shows a learning-to-rank component that depends on a BM25 retrieval and on uploaded query features (e.g., user behavior data like clicks and dwell-times obtained from a user study). When a component has inputs from more than one component, TIRA makes them available in the order in which the components have been defined. Preceding components or uploads must exist when defining a new component and TIRA decouples the command to be executed from the Docker image so that the same image can be used to run different retrieval approaches (e.g., by switching parameters). In combination with caching, this improves the efficiency for a wide range of common multi-stage retrieval pipelines.

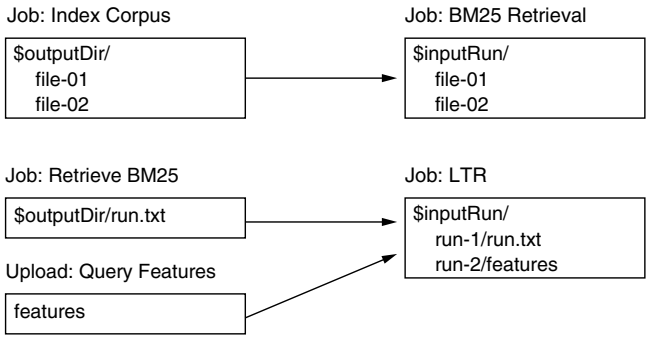

Figure 2: Data flow of two retrieval pipelines in TIRA. The upper retrieval pipeline creates an index so that the second stage retrieves from the index with BM25. The bottom retrieval pipeline uses a BM25 ranking and a manually uploaded file with query features as input for an LTR algorithm.

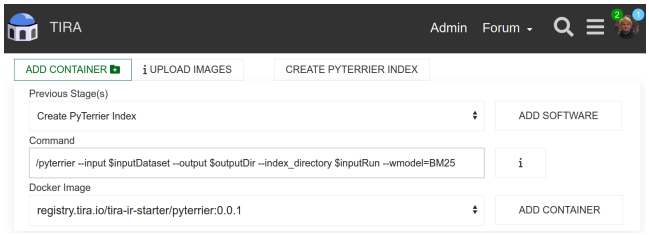

Figure 3: Defining a BM25 retrieval component in TIRA that depends on a previously created PyTerrier index.

Efficiency via Caching. As every software in TIRA is immutable, coding errors in a component can only be fixed by adding a new

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

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

Listing 2: Re-ranking BM25 with a submitted software.

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

version of the component. Immutability enables the implementation of efficient and reliable retrieval pipelines, since their output can both be cached and traced back and replicated by that same (version) of a component. TIRA disallows the deletion of components or outputs that have been used as inputs by some other component. When any component is requested to produce an output on some data, it is first checked whether that output already is cached in which case executing the component is not necessary. This way, retrieval pipelines in TIRA can efficiently re-use components and remain replicable, as the steps to produce a final run are fully tracked and versioned in the experiment repository.

3.6 Local Pipeline Reproduction with PyTerrier

When an experiment repository is exported and published by the organizers, by default, the test data is kept private but the run files are published via TIRA and software submissions are uploaded as Docker images to Docker Hub. All possible follow-up studies (e.g., a reproducibility study for a shared task) can be conducted independent of TIRA, as archived experiment repositories are fully self-contained. In the following, we briefly showcase some post-hoc experiments in PyTerrier.[11]

Listing 1 shows how a full-rank approach from a TIRA experiment repository can be reproduced with a declarative PyTerrier pipeline. The approach is identified as the `<software>` submitted by team `<user-name>` to the shared task `<task-name>` and is applied to `<dataset>` (does not need to be the original task data). Internally, the required Docker images are downloaded and run in their required order to obtain the results. These results can then be re-ranked by any PyTerrier re-ranker, allowing for experiments to improve an original submission. Also re-rankers available in some TIRA experiment repository can be used in post-hoc PyTerrier experiments (cf. Listing 2 for an example re-ranking of BM25).

Listing 3 shows how run files resulting from some (software) submission can be loaded into PyTerrier. The `from_submission` method allows to access some submitted approach's output without having to re-run it (e.g., this also eases pooling for task organizers). The PyTerrier integration allows easy replicability experiments if the dataset is the same as in the original experiment, and reproducibility experiments if some other dataset is used for retrieval approaches.

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

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

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

Table 3: The 15 corpora and the associated 32 retrieval tasks currently available in TIREx (all are open for submissions).

Corpus			Associated Retrieval Tasks	
Name	Docs.	Size	Details	#
Args.me	0.4 m	8.3 GB	Touché 2020–2021 [8, 11]	2
Antique	0.4 m	90.0 MB	QA Benchmark [50]	1
ClueWeb09	1.0 b	4.0 TB	Web tracks 2009–2012 [23–26]	4
ClueWeb12	731.7 m	4.5 TB	Web tracks [30, 31], Touché [10, 11]	4
ClueWeb22B	200.0 m	6.8 TB	Touché 2023 [9] (ongoing)	1
CORD-19	0.2 m	7.1 GB	TREC-COVID [93, 98]	1
Cranfield	1,400	0.5 MB	Fully Judged Corpus [28, 29]	1
Disks4+5	0.5 m	602.5 GB	TREC-7/8 [95, 96], Robust04 [89, 90]	3
GOV	1.2 m	4.6 GB	Web tracks 2002–2004 [33–35]	3
GOV2	25.2 m	87.1 GB	TREC TB 2004–2006 [19, 22, 27]	3
MEDLINE	3.7 m	5.1 GB	TREC Genomics [51, 52], PM [81, 82]	4
MS MARCO	8.8 m	2.9 GB	Deep Learning 2019–2020 [36, 37]	2
NFCorpus	3,633	30.0 MB	Medical LTR Benchmark [13]	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

4 EVALUATION

To demonstrate the scalability of TIREx, we report about an experiment with 50 retrieval approaches on 32 retrieval tasks based on 15 corpora (1.9 billion documents). The resulting leaderboards are public and new submissions can be made at any time.[12] We also describe a `repro_eval`-based [16] case study on system preference reproducibility for different tasks.

4.1 Scalable Retrieval Experiments

Table 3 shows the 15 corpora currently available in TIREx. Each has been used for 1 to 4 shared retrieval tasks, consists of 1,400 to 1 billion documents, and comes with the relevance judgments created during the respective shared tasks.

Table 4 overviews the 50 retrieval approaches that we imported into TIREx from 5 retrieval frameworks: BEIR [86], ChatNoir [7], Pyserini [64] (our import was not ready during the experiments), PyGaggle [64], PyTerrier [69] (including two PyTerrier plugins for duoT5 [79] and ColBERT [59]). From BEIR, we use 17 dense retrieval approaches (e.g., ANCE [99], DPR [58], and TAS-B [53]) by using the different SBERT [80] models available in BEIR. ChatNoir is an Elasticsearch-based BM25F search engine hosting all three ClueWeb corpora. It can be accessed from within TIRA to allow retrieval approaches on huge corpora with a REST-API that is kept consistent to ensure reproducibility. From Pyserini, we use the 4 lexical models available trough the SimpleSearcher interface. From PyGaggle, we

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

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

Framework	Type	Description	Approaches	
			Full-rank	Re-rank
BEIR [86]	Bi-encoder	Dense retrieval	17	17
ChatNoir [7]	BM25F	Elasticsearch cluster	1	0
ColBERT@PT [59]	Late interaction	PyTerrier plugin	0	1
DuoT5@PT [79]	Cross-encoder	Pairwise transformer	0	3
PyGaggle [64]	Cross-encoder	Pointwise transformer	0	8
PyTerrier [69]	Lexical	Traditional baselines	20	20
Pyserini* [64]	Lexical	Traditional baselines	4	4

*Our import of Pyserini was not ready during the experiments but is now available.

use 8 variants of monoBERT [74] and monoT5 [79] (including the state-of-the-art monoT5 with 3 billion parameters), and from PyTerrier, we use 20 lexical retrieval models (e.g., BM25, PL2, etc.). From the duoT5 plugin of PyTerrier, we use 3 variants based on different duoT5 models (including the state-of-the-art model with 3 billion parameters). For all retrieval approaches, we keep all parameters at their default values. Almost all approaches use the default_text-based fields that we added to `ir_datasets`, except for ChatNoir that is a full-rank software for the ClueWeb corpora and uses different fields (title, body, etc.). The lexical approaches in PyTerrier and the dense approaches in BEIR can be configured as full-rank software (i.e., a first component building an index and a second component retrieving from the index) or re-rank software—but are just counted as one approach in Table 4. All duoT5 and PyGaggle approaches only work as re-rankers. For ColBERT, we only use the re-rank variant, as ColBERT indices become very large.

In TIREx, all of these variants are available. To increase result comparability, however, our analysis fixes the first stage rankers to ChatNoir for the ClueWeb corpora and PyTerrier BM25 on all other corpora. Their respective results are then handed to the total of 50 available re-ranking approaches mentioned above. Altogether, 50 approaches are executed on all 32 tasks listed in Table 3. We executed the lexical approaches using 1 CPU and 10 GB RAM, while all other approaches had additional access to a GeForce GTX 1080 GPU with 8 GB RAM. Some models fail on this GPU as 8 GB of RAM do not suffice: ColBERT and two SBERT models failed on a few tasks, while the 3 billion parameter monoT5 / duoT5 failed on all tasks. To handle these cases, we added two runners with access to an A100 GPU with 40 GB RAM to TIRA, which was sufficient. TIRA manages metadata about the resources used to produce a run, making hardware difference between evaluations transparent.

Table 5 shows the aggregated evaluation results on 31 tasks (leaving out the ClueWeb22 as there are no judgments yet). We report the effectiveness as nDCG@10 (macro-averaged in case a corpus is associated with multiple tasks) for BM25, ColBERT, TAS-B, all three duoT5 variants, and monoT5 (in its default configuration with its default model) and the best, median, and worst approaches from the groups of 20 lexical, 17 bi-encoder, and 8 PyGaggle approaches. All deep learning models were trained on MS MARCO and thus substantially improve upon the lexical models on MS MARCO. However, on other corpora the deep learning models work in a zero-shot manner so that sometimes a lexical approach achieves the highest effectiveness (Args.me, ClueWeb09, and MEDLINE). Our results further show that BM25 is not always the best lexical ranker (e.g., on Args.me: 0.43 vs. 0.57). The effectiveness gap between the best and the worst model of a group can be substantial on some corpora (e.g., lexical models on Args.me: 0.14 vs. 0.57), while being negligible on others (e.g., lexical models on NFCorpus). The leaderboards of TIREx as aggregated in Table 5 allow to easily select competitive baselines for very different tasks—often much easier than before.

4.2 Case Study: Reproducibility Analysis

As an example of a post-hoc analysis enabled by TIREx, we use `repro_eval` to analyze to which degree system preferences from the TREC Deep Learning 2019 task can be reproduced on other tasks. For each preference between approaches on TREC Deep Learning 2019 (e.g., monoT5 with an nDCG@10 of 0.71 compared to BM25's 0.48 induces a clear system preference), we set the approach with the lower effectiveness on TREC Deep Learning 2019 as the "baseline" in `repro_eval` and the other approach as the "advanced system". We study the reproducibility of the preferences on two dimensions [15]: (1) the effect ratio of the reproduction, and (2) the delta relative improvement of the reproduction. The effect ratio measures to which degree the advanced system is still better than the baseline on the different task (1 indicates a perfect reproducibility, values between 0 and 1 indicate reproducibility with diminished improvements on the different task, and 0 indicates failed reproducibility), while the delta relative improvement measures the relative effectiveness difference of the advanced system to the baseline (0 indicates perfect reproducibility, values between -1 and 0 indicate an increased relative improvement of the advanced system, values between 0 and 1 indicate a smaller relative improvement, and 1 indicates failed reproducibility).

Table 6 shows the results of the preference reproducibility analysis. We report the ratio of system preferences with a successful reproduction (i.e., effect ratio > 0) and the 25%, 50%, and 75% quantiles for the effect ratio and the relative delta improvement. We order the tasks by the percentage of successfully reproduced preferences and show the top-5 tasks and every fifth lower ranked task. Not that surprising, the reproducibility on the very similar TREC Deep Learning 2020 is very good (88.1%) but declines fast for other tasks (e.g., only 57.8% for the Web track 2003 on rank 15). Analyzing 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 Deep Learning 2020, while the Web track 2003 on rank 15 has a median effect ratio of 0.04).

5 DISCUSSION

Potential Impact of TIREx. We believe that TIREx can have a substantial conceptual impact as we see no alternative to blinded retrieval evaluations in the future (given the practice of training LLMs on basically all available ground truth for IR and NLP tasks [20]). Additionally, the platform eases the organization of reproducible IR experiments with software submissions. Shared task organizers can simply provide the well-documented open-source baselines from TIRA as starting points for the participants and can also use the baselines to ensure some more diverse judgment pools, especially for tasks that attract few participants. For shared tasks that

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 TREC DL 2019 system preferences on other tasks. Success rate in percent (effect ratio > 0; tasks ordered by success rate) and the 25%, 50%, and 75% quantiles for the effect ratio and delta relative improvement.

Task	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

run multiple years on different data, the organizers can automatically re-run all approaches submitted to previous editions to track progress. TIREx combines leaderboards with immutable software, promoting provenance of results, and enabling researchers and reviewers to identify and locally reproduce good baselines.

The submission platform TIRA proved robust after its complete redevelopment [43]: two NLP tasks used TIRA at SemEval 2023 [42, 60] for which 71 of the 171 registered teams created 647 runs with software submissions. Our initial retrieval experiments with TIREx produced another 1,600 runs on standard corpora in less than a week, showing the platform to be robust and to have the potential for scaling up. When adopted by shared tasks and in individual IR experiments, TIREx can become a (federated) hub for IR resources and serve as a reference for reviewers. If a sufficient number of retrieval approaches, corpora, and supplementary data (e.g., manual query reformulations) are available through TIREx, integrating new resources gives direct access to an entire ecosystem, furthering the nascent standardization of IR experiments.

Future Extensions of TIREx. Interesting directions for future development besides including further IR frameworks and libraries are integrations of TIREx with the IR Anthology [78] and with DiffIR [57]. An integration with the IR Anthology would enable links between entries in the TIREx leaderboards and the corresponding publications in the IR Anthology to provide more detailed information on an approach but also to "extend" a publication by adding results on different corpora than originally used and putting an approach in a broader context with other approaches run on the same data. An integration with DiffIR would enable the rendering of runs as search engine result pages to easily contrast the quantitative evaluations already possible via the integrated `ir_measures` with more qualitative evaluations of ranking differences or even (basic) user studies.

6 CONCLUSION

With TIREx—The IR Experiment Platform—we aim to substantially ease conducting (blinded) IR experiments and organizing "always-on" reproducible shared tasks on the basis of software submissions. TIREx integrates `ir_datasets`, `ir_measures`, and PyTerrier with TIRA. Retrieval workflows can be executed on-demand via cloud-native orchestration, reducing the effort for reproducing IR experiments since software submitted to TIREx can be re-executed in post-hoc experiments. The platform has no lock-in effect, as archived experiments are fully self-contained, work standalone, and are easily exported. By keeping test data private, TIREx promotes further standardization and provenance of IR experiments following the example of, e.g., medicine, where blinded experiments are the norm. TIREx is open to the IR community and ready to include more corpora, shared tasks, 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, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, and M. Hagen. Overview of Touché 2020: Argument retrieval. *CLEF 2020*. 384–395.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

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

[18] 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.

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

[20] 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).

[21] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[38] 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.

[39] 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.

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

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

[42] M. Fröbe, T. Gollub, M. Hagen, and M. Potthast. SemEval-2023 task 5: Clickbait spoiling. *SemEval-2023*. 2278–2289.

[43] 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.

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

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

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

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

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

[49] M. Hagen, M. Potthast, and B. Stein. Overview of the author obfuscation task at PAN 2017: Safety evaluation revisited. *CLEF 2017*. 1613–0073.

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

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

[52] 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*.

[53] 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.

[54] 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.

[55] F. Hopfgartner, A. Hanbury, H. Müller, I. Eggel, K. Balog, T. Brodt, G. V. Cormack, J. Lin, J. Kalpathy-Cramer, N. Kando, M. 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.

[56] 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.

[57] K. M. Jose, T. Nguyen, S. MacAvaney, J. Dalton, and A. Yates. DiffIR: Exploring differences in ranking models' behavior. *SIGIR 2021*. 2595–2599.

[58] 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.

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

[60] J. Kiesel, M. Alshomary, N. Mirzakhmedova, M. Heinrich, N. Handke, H. Wachsmuth, and B. Stein. SemEval-2023 task 4: ValueEval: Identification of human values behind arguments. *SemEval-2023*. 2290–2306.

[61] 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.

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

[63] 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.

[64] 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.

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

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

[67] 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.

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

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

[70] C. Macdonald and N. Tonellotto. Declarative experimentation in information retrieval using PyTerrier. *ICTIR 2020*. 161–168.

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

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

[73] 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*.

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

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

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

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

[78] M. Potthast, S. Günther, J. Bevendorff, J. P. Bittner, A. Bondarenko, M. Fröbe, C. Kahmann, A. Niekler, M. Völske, B. Stein, and M. Hagen. The information retrieval anthology. *SIGIR 2021*. 2550–2555.

[79] 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).

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

[81] 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*.

[82] 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*.

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

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

[85] 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*.

[86] 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*.

[87] 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.

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

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

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

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

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

[93] 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.

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

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

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

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

[98] 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).

[99] 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*.

[100] 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).

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

[102] 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.

[103] 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).

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