
There may be differences between this version and the published version. You are advised to consult the published version if you wish to cite from it.

https://eprints.gla.ac.uk/296336/

Deposited on 2 May 2023
We integrate \texttt{ir\_datasets}, \texttt{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 \texttt{ir\_datasets} and \texttt{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 · 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.

### 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.\footnote{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.} In any case, the current best practices for shared tasks do not enforce “blinded experimentation”\footnote{en.wikipedia.org/wiki/Blinded\_experiment} 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,\footnote{github.com/tira-io/ir-experiment-platform} a key feature of TIREx is the full integration
We review ad hoc retrieval experiments in evaluation campaigns, that the pools are “essentially complete”, i.e., unjudged documents evaluations on the same corpus usually are based on the assumption 
submitted systems’ top-ranked documents [85]. Subsequent eval-
immediately thereafter [85]. The current practice at shared tasks 
grew substantially, complete judgments became infeasible almost 
complete relevance judgments for 225 topics. Since corpus sizes 
ments [27, 28] were conducted on a corpus of 1,400 documents with 
the Cranfield experiments [85]. In the 1960s, the Cranfield experi-
ments [27, 28] were conducted on a corpus of 1.9 billion documents). This 1,600-runs experiment was started by just 
new approach against weak / “wrong” baselines (i.e., not the best or 
diversity data is not much effort. In TIREx can rather easily be evaluated against many / all corpora 
help to further improve both: the internal validity via archiving 
all experiments and results on some corpus (e.g., to accurately cor-
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 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” exper-
iments. 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 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.

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 experi-
ments [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 eval-
uations 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 comple-
teness 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” exper-
iments. 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 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.

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
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 conferences6 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.

### 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

---

5http://www.netflixprize.com/


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 Touche\(^10\) 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 Touche 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 Touche retrieves against the ClueWeb22, which is made available in TIRA through ChatNoir \([7]\), but retrieving the top-1000 results for 50 Touche \([8–10]\) 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
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 \texttt{ir_datasets} Docker image and its configuration. We provide standard Docker images that organizers can use if their dataset is already available in \texttt{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 \texttt{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 \cite{35, 36, 67} provides a single text field, while \texttt{Args.me} \cite{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, \texttt{Antique} \cite{47} has a single text query field, while TREC Precision Medicine \cite{74, 75} provides multiple fields.

TIRA automatically evaluates all runs created by software submissions or uploads. We provide an \texttt{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).

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 \texttt{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 \texttt{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 \texttt{ir_datasets} integration makes available to retrieval software. For full-rank software, the \texttt{ir_datasets} integration creates a documents.tar.gz file that contains the document identifier, the default text of the document, and all structured fields of the document and a file topics.tar.gz with the topics, both in JSON Lines format. For re-rankers, the \texttt{ir_datasets} integration creates, given an arbitrary run file obtained from the previous stage, a file \texttt{re-rank.tar.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 \texttt{ir_datasets} integration is executed before the re-ranker and the re-ranker only gets the \texttt{re-rank.tar.gz} file as input. Furthermore, the \texttt{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 \texttt{ir_measures}

TIRA automatically evaluates all runs created by software submissions or uploads. We provide an \texttt{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-rank 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).

<table>
<thead>
<tr>
<th>Type</th>
<th>Resource</th>
<th>Fields</th>
<th>Access</th>
<th>Example Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Full-Rank</td>
<td>documents.jsonl.gz</td>
<td>docno, text, original_document</td>
<td>✓</td>
<td>['docno': '8182161', 'text': 'Goldfish can grow up to 18 inches …', 'original_document': [...]]</td>
</tr>
<tr>
<td></td>
<td>topics.jsonl.gz</td>
<td>qid, query, original_topic</td>
<td>✓</td>
<td>({'qid': '156493', 'query': 'do goldfish grow', 'original_query': [...]})</td>
</tr>
<tr>
<td>Re-Rank</td>
<td>re-rank.jsonl.gz</td>
<td>qid, query, original_topic, docno, text, original_document, score, rank</td>
<td>✓</td>
<td>({'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})</td>
</tr>
<tr>
<td>Both</td>
<td>qrels.txt</td>
<td>topic, iteration, docno, relevance</td>
<td>✗</td>
<td>156493 Q0 8182161 2</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Availability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$inputDataset</td>
<td>Always</td>
<td>Directory containing the input data.</td>
</tr>
<tr>
<td>$outputDir</td>
<td>Always</td>
<td>Directory with expected output data.</td>
</tr>
<tr>
<td>$inputRun</td>
<td>Multi-Comp.</td>
<td>Output(s) of previous stage(s).</td>
</tr>
</tbody>
</table>

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 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 the subsequent stages as much as possible to keep the overall software replicable.

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
work stand-alone, as the archived repository is fully self-contained. Was used by some other component as input cannot be deleted. After the experiment repository is exported and published by the experiment organizers, the submitted retrieval approaches and the produced judgment pools. Altogether, the PyTerrier integration allows easy reproducibility (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-ranking approaches.

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 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.

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

```python
dataset = <task-name>/<user-name>/software

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

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

```python
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.

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

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. 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 CoBERT [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.

---

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

12https://github.com/tira-io/ir-experiment-platform#submission
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 produce a run, making differences transparent. 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 3: The 15 corpora and the associated 32 retrieval tasks currently available in TIREx (submission possible).

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Associated Retrieval Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Docs.</td>
</tr>
<tr>
<td>Args.me</td>
<td>0.4m</td>
</tr>
<tr>
<td>Antique</td>
<td>0.4m</td>
</tr>
<tr>
<td>ClueWeb12</td>
<td>731.7m</td>
</tr>
<tr>
<td>ClueWeb2B</td>
<td>200.0m</td>
</tr>
<tr>
<td>CORD-19</td>
<td>0.2m</td>
</tr>
<tr>
<td>Cranfield</td>
<td>1.400</td>
</tr>
<tr>
<td>DiskS4+5</td>
<td>0.5m</td>
</tr>
<tr>
<td>GOV</td>
<td>1.2m</td>
</tr>
<tr>
<td>GOV</td>
<td>25.2m</td>
</tr>
<tr>
<td>MEDLINE</td>
<td>3.7m</td>
</tr>
<tr>
<td>MS MARCO</td>
<td>8.8m</td>
</tr>
<tr>
<td>NFCorpus</td>
<td>3.633</td>
</tr>
<tr>
<td>Vasewani</td>
<td>11,429</td>
</tr>
<tr>
<td>WaPo</td>
<td>0.6m</td>
</tr>
<tr>
<td>Sum = 15 corpora</td>
<td>1.9b</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Framework</th>
<th>Type</th>
<th>Description</th>
<th>Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEIR [79]</td>
<td>Bi-encoder</td>
<td>Dense retrieval</td>
<td>17</td>
</tr>
<tr>
<td>ChatNoir [7]</td>
<td>BM25F</td>
<td>Elasticsearch cluster</td>
<td>1</td>
</tr>
<tr>
<td>ColBERT@PT [55]</td>
<td>Late interaction</td>
<td>PyTerrier plugin</td>
<td>1</td>
</tr>
<tr>
<td>Duot5@PT [72]</td>
<td>Cross-encoder</td>
<td>Pairwise transformer</td>
<td>3</td>
</tr>
<tr>
<td>PyGaggle [59]</td>
<td>Cross-encoder</td>
<td>Pointwise transformer</td>
<td>8</td>
</tr>
<tr>
<td>PyTerrier [64]</td>
<td>Lexical</td>
<td>Traditional baselines</td>
<td>20</td>
</tr>
</tbody>
</table>

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, and values between 0 and 1 indicate a smaller relative improvement, and 1 indicates a failed reproduction).
We sort the tasks by the percentage of successful reproductions.

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).

<table>
<thead>
<tr>
<th>Corpus</th>
<th>ChatNoir</th>
<th>Lexical</th>
<th>Late Int.</th>
<th>Bi-Encoder</th>
<th>duoT5</th>
<th>PyGaggle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antique</td>
<td>—</td>
<td>0.51</td>
<td>0.53</td>
<td>0.51</td>
<td>0.56</td>
<td>0.47</td>
</tr>
<tr>
<td>Arg5one</td>
<td>—</td>
<td>0.43</td>
<td>0.37</td>
<td>0.43</td>
<td>0.14</td>
<td>0.26</td>
</tr>
<tr>
<td>CORD-19</td>
<td>—</td>
<td>0.28</td>
<td>0.64</td>
<td>0.35</td>
<td>0.21</td>
<td>0.58</td>
</tr>
<tr>
<td>ClueWeb09</td>
<td>0.16</td>
<td>0.18</td>
<td>0.24</td>
<td>0.18</td>
<td>0.12</td>
<td>0.17</td>
</tr>
<tr>
<td>ClueWeb12</td>
<td>0.36</td>
<td>0.24</td>
<td>0.27</td>
<td>0.25</td>
<td>0.14</td>
<td>0.23</td>
</tr>
<tr>
<td>Cranfield</td>
<td>—</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Disks4+5</td>
<td>—</td>
<td>0.44</td>
<td>0.46</td>
<td>0.37</td>
<td>0.36</td>
<td>0.39</td>
</tr>
<tr>
<td>GO</td>
<td>—</td>
<td>0.22</td>
<td>0.24</td>
<td>0.22</td>
<td>0.15</td>
<td>0.23</td>
</tr>
<tr>
<td>GOV2</td>
<td>—</td>
<td>0.47</td>
<td>0.49</td>
<td>0.44</td>
<td>0.25</td>
<td>0.45</td>
</tr>
<tr>
<td>MS MARCO</td>
<td>—</td>
<td>0.49</td>
<td>0.50</td>
<td>0.48</td>
<td>0.57</td>
<td>0.69</td>
</tr>
<tr>
<td>MEDLINE</td>
<td>—</td>
<td>0.34</td>
<td>0.42</td>
<td>0.27</td>
<td>0.18</td>
<td>0.25</td>
</tr>
<tr>
<td>NFCorpus</td>
<td>—</td>
<td>0.27</td>
<td>0.28</td>
<td>0.27</td>
<td>0.26</td>
<td>0.27</td>
</tr>
<tr>
<td>Vaswani</td>
<td>—</td>
<td>0.45</td>
<td>0.46</td>
<td>0.45</td>
<td>0.30</td>
<td>0.43</td>
</tr>
<tr>
<td>WaPo</td>
<td>—</td>
<td>0.38</td>
<td>0.39</td>
<td>0.37</td>
<td>0.24</td>
<td>0.43</td>
</tr>
<tr>
<td>Avg.</td>
<td>—</td>
<td>0.34</td>
<td>0.39</td>
<td>0.35</td>
<td>0.22</td>
<td>0.33</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>25% 50% 75%</td>
<td>25% 50% 75%</td>
</tr>
<tr>
<td>TREC DL 2020</td>
<td>1</td>
<td>88.1</td>
<td>0.68 0.90 1.11</td>
<td>-0.03 0.02 0.08</td>
</tr>
<tr>
<td>Touché 2020 (Task 2)</td>
<td>2</td>
<td>77.1</td>
<td>0.12 0.38 0.73</td>
<td>-0.09 0.04 0.17</td>
</tr>
<tr>
<td>Web Track 2004</td>
<td>3</td>
<td>75.5</td>
<td>0.01 0.29 0.89</td>
<td>-0.07 0.10 0.31</td>
</tr>
<tr>
<td>TREC-7</td>
<td>4</td>
<td>73.9</td>
<td>0.01 0.01 0.11</td>
<td>-0.02 0.12 0.34</td>
</tr>
<tr>
<td>Core 2018</td>
<td>5</td>
<td>70.2</td>
<td>0.05 0.24 0.90</td>
<td>-0.03 0.13 0.35</td>
</tr>
<tr>
<td>NFCorpus</td>
<td>10</td>
<td>66.4</td>
<td>-0.06 0.06 0.32</td>
<td>0.02 0.23 0.42</td>
</tr>
<tr>
<td>Web track 2003</td>
<td>15</td>
<td>57.8</td>
<td>-0.14 0.04 0.23</td>
<td>-0.08 0.15 0.36</td>
</tr>
<tr>
<td>Web track 2009</td>
<td>20</td>
<td>44.1</td>
<td>-0.04 0.04 0.26</td>
<td>0.00 0.30 0.52</td>
</tr>
<tr>
<td>Web track 2010</td>
<td>25</td>
<td>36.3</td>
<td>-0.49</td>
<td>-0.14 0.18</td>
</tr>
<tr>
<td>Web track 2013</td>
<td>30</td>
<td>31.0</td>
<td>-0.43</td>
<td>-0.21 0.13</td>
</tr>
</tbody>
</table>

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 TIREx 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).


