Leipzig University
Institute of Computer Science
Degree Programme Computer Science, B.Sc.

Improving the Validity of Information Retrieval Experiments by Making Test Collections Exchangeable

Bachelor's Thesis

Simon Reich

1. Referee: Prof. Dr. Martin Potthast

Submission date: September 9, 2023

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Leipzig, September 9, 2023

.....
Simon Reich

Abstract

We integrate `ir_datasets`, PyTerrier, and `ir_measures` into TIRA to improve the validity of information retrieval experiments by facilitating replicability, thereby promoting large-scale experiments that advance research toward generalization. `ir_datasets` is wrapped inside the `ir_datasets_loader` to load and distribute standardized data for full-ranking and re-ranking tasks, compatible with the PyTerrier interface. This enables the execution of fully modularized retrieval pipelines, making both retrieval software and data sets interchangable. We leverage this implementation to evaluate 50 retrieval systems on 31 test collections, ultimately performing three post-hoc studies on the yielded data. First, we evaluate the replicability of system preferences on Deep Learning track 2019 across 30 test collections. Second, we deploy feature-rich learning-to-rank pipelines with 43 features on the Clueweb 2009-1014 test collections. Lastly, we conduct a comparative study of rank-fusion with the results of 49 retrieval systems on 25 test collections.

Contents

Acknowledgements

I would like to thank Maik Fröbe and Jan Heinrich Reimer for their invaluable support and supervision throughout the development of this thesis. The conceptual foundation and structural framework of The Information Retrieval Experiment Platform, as presented in this thesis, are their original contributions. I would also like to specifically thank Maik Fröbe for implementing all retrieval software and test collections into TIREx, which provided me with the data for all post-hoc experiments discussed herein.

Chapter 1

Introduction

Information Retrieval is primarily a empirical science in which knowledge is gained mostly through experimentation [2]. As in any science, it is highly desirable that their scientific findings be valid. However, to ensure the validity of its scientific claims, the experiments on which they are based must be validly conducted in the first place.

Experiments in information retrieval are often conducted on carefully constructed test collections. The blueprint for this methodology dates back to the 1960s, when the Cranfield experiments [29, 30] established a robust method for evaluating the performance of a retrieval approach [90]. The retrieval system is implemented in a controlled test environment with a fixed document corpus and a fixed topic list. It generates ranked lists of documents, known as "runs", based on their relevance to each topic. For evaluation, a set of relevance judgments is provided, which lists the relevance of the documents towards the topics, allowing the system's effectiveness to be measured using metrics like Precision (P) [13] or Normalized Discounted Cumulative Gain (nDCG) [60]. This standardized approach also enables the comparison of different retrieval systems on the same test collection [88].

Today, hundreds of test collections are publicly available to the community. Test collections often originate from information retrieval evaluation campaigns [88], such as TREC[1], CLEF[2], and others. Research at these conferences is usually organized as a shared task: Organizers provide a specific task, along with a data set, containing both, documents and topics. Participants implement their retrieval algorithms on the data set and submit their generated run files back to the organizers. The submitted runs are then pooled, evaluated, and used to create a set of relevance judgments [90], enabling evaluation of retrieval systems on that test collection. Released test collections are frequently reused

[1] `https://trec.nist.gov/`

[2] `https://clef2022.clef-initiative.eu/`

in subsequent information retrieval research, particularly in small-scale, independent laboratory experiments. The results of these experiments are then published within the scientific community, further contributing to an ongoing progression of the field.

This mélange of resourceful evaluation campaigns, manufacturing expensive but necessary artifacts for widespread laboratory research, which then again feedbacks to the community, definitely had it's success in pushing the field as a whole. But the way research is often conducted has faced increasing criticism from parts of the community. In 2009, Armstrong et al. published their investigation into the effectiveness of information retrieval research [2], suggesting that progress in the field may have subtly stagnated due to the widespread use of weak baselines. This issue has been revisited several times, renewing the claim [62, 65, 98], and additionally a variety of other shortcomings and pitfalls in information retrieval research emerged over the years. These include a critical lack of reproducibility of information retrieval experiments as several studies report on failed attempts to reexecute experiments [43, 64, 76, 93], even when they were documented in computational notebooks [76]. The unrestricted accessibility of data, on the other hand, has the potential to bias research hypotheses [77]. Even evaluation campaigns still distribute data sets instead of collecting software to run them in restricted environments [53]. And last but not least, there still seems to be a particularly low investment in generalization testing, resulting in an overall thin theoretical foundation of the field [44, 47, 49].

The work behind this thesis contributed to **T**he **I**nformation **R**etrieval **Ex**periment Platform (TIREx) [45], a retrieval framework conceptualized and developed by the Webis group[3] to tackle all of these challenges at once, specifically designed to improve the validity of information retrieval experiments. TIREx is built on TIRA [45, 77], an Evaluation-as-a-Service platform which already enables blind testing through the Algorithm-to-Data paradigm. In addition, TIRA facilitates reproducibility by integrating Docker and Git and enables archiving of information retrieval experiments in fully self-contained Git repositories. By further integrating three popular IR tools, namely `ir_datasets` [71], PyTerrier [72], and `ir_measures`[4], into TIRA, we introduce a standardized I/O interface to retrieval pipelines within the framework. This enables TIREx to execute fully modularized retrieval pipelines. Once test collections and retrieval systems are implemented, they become interchangeable by default. This achievement contributes to the effort to drive research towards generalization, as replicating previous experiments on different test collections becomes easier

[3] `https://webis.de/`

[4] `https://github.com/terrierteam/ir_measures`

than ever before.

While Chapter 3 provides details about the implementation of TIREx, Chapter 4 showcases the potential of the framework to facilitate large scale investigations by evaluating three post-hoc experiments. After the implementation and evaluation of 50 retrieval systems on 31 test collections, resulting in a total of 1550 runs, we investigate the replicability of system preferences between all 50 retrieval systems on the TREC Deep Learning Track 2019 [37] across all other 30 test collections. We further evaluate 6 feature-rich learning-to-rank pipelines with features from 43 retrieval systems derived from the data gathered with TIREx. These pipelines are seamlessly implemented by the PyTerrier implementation. Finally, we evaluate rank-fusion between the runs of 49 retrieval systems on 25 test collections. The thesis closes with a concluding summary in Chapter 5.

Chapter 2

Background and Related Works

In this chapter, I provide an overview of the origins and best practices of contemporary experimental frameworks in information retrieval research, review them in terms of common validity criteria of empirical science, and introduce contemporary concepts and software designed to support the validity of information retrieval experiments.

2.1 Experiments in Information Retrieval

Research in information retrieval has been strongly empirical [2], at least since the advent of the Cranfield paradigm, which manifested basic experimental evaluation methodologies that are still in use today [90]. Since the early 1990s, research and development is fueled by collaborative workshops, which yield resourceful artifacts to further foster individual laboratory research that continually comment on and complement to the achievements made.

The Cranfield Paradigm. The advent of computer systems spurred new possibilities of information storage but lacked a scientific approach to leverage information retrieval effectively [29]. To gain a more systematic understanding, the Cranfield experiments of the 1960s [29, 30] were among the first to introduce the concept of controlled, laboratory-style experiments in the field of information retrieval, aiming to standardize methodologies and develop objective performance measures for different retrieval algorithms and techniques [90]. To simulate a form of searching large amounts of textual data by short queries and to measure the effectiveness of retrieval, a data set is used consisting of a fixed document corpus, a set of topics, i.e., the search queries, and a set of relevance judgments containing information about the relevance of the documents to the topics. To measure the performance of a retrieval system, the documents are indexed and then ranked by the system, based on

its computed predictions about their relevance to the topics. For each topic the system outputs a "run file", which lists the documents in order of their predicted relevance score. The run file is then evaluated by the use of the judgments and a metric of choice, originally Precision and Recall[1], which computes the effectiveness of the system towards the given topic. Finally, to measure the systems overall performance on the test collection, an average is computed across all evaluated run files.

Although this concept is based on assumptions which are not strictly true, e.g., a static collection of documents or the assumption of binary notion of relevance and its independence from the user's context [88], Voorhees et al. showed that the methodology has proven to be insightful and robust over time [90]. Conserving its efficiency in performance measurement and comparison, the framework turned out to be adaptable to different tasks [88, 90], still forming the basis for most of today's information retrieval experiments [90].

While the pioneering Cranfield test collection consists of 1.400 documents, 225 topics and a complete set of relevance judgments, today's collections are usually far greater in size, containing up to millions of documents and an incomplete assessment of relevance, due to the fact that relevance judgment has to be manufactured by humans resources [90]. Contemporary test collections are usually a product of shared tasks at information retrieval evaluation campaigns.

Research at Evaluation Campaigns. Substantial experimental research in information retrieval is carried out as annual shared efforts, i.e., *Tracks*, at international conferences, at least since 1992 with the foundation of the **TE**xt **R**etrieval **C**onference (TREC)[2]. Since then, the concept has spread and many spin-offs derivated, such as the **N**II **T**estbeds and **C**ommunity for **I**nformation access **R**esearch (NTCIR)[3], the **C**onference and **L**abs of the **E**valuation **F**orum (CLEF)[4] and the **F**orum for **I**nformation **R**etrieval **E**valuation (FIRE)[5], among many others.

Over the years and with the continued exploration of the field, as well as advance in technologies and application needs, different domains of retrieval challenges have been identified and elaborated at these events. These challenges cover a broad spectrum of retrieval tasks such as ad-hoc retrieval, passage retrieval, argument retrieval, question answering, entity retrieval, and many more. Among these, ad-hoc retrieval is the oldest and most constant one, rep-

[1] `https://en.wikipedia.org/wiki/Precision_and_recall`
[2] `https://trec.nist.gov/`
[3] `https://research.nii.ac.jp/ntcir/index-en.html`
[4] `https://clef2022.clef-initiative.eu/`
[5] `http://fire.irsi.res.in/`

resenting a traditional retrieval problem of searching a set of documents with a query to retrieve a relevant subset.

Research at these conferences is typically organized as shared tasks, each addressing a specific retrieval challenge. The organizers set the task and distribute a new data set, which consists of a set of documents and a set of topics. Participants then apply their retrieval approaches onto the test collection to solve the task. After running their system on the test data, they submit their results, which usually include the run files their systems produced. These run files are used by the organizers to craft a set of relevance judgments by 'pooling' the top-ranked documents from each participant's submissions, evolving the data set into a new test collection. Pooling ensures a fair and comprehensive assessment across all contributing systems while determine the subset of documents to assess [90]. For assessment human agents determine the relevance of these pooled documents to the corresponding topics. Once the judgments are in place, organizers can evaluate each participant's retrieval approach, measuring their performance and comparing them with each other. Test collections constructed at evaluation campaigns are usually published soon after and eventually reused frequently by the community in individual laboratory research.

Experimental Laboratory Research in Information Retrieval. Naturally, there is no limit to topics, methodologies and scale of individual research outside the realm of resourceful shared tasks. In this thesis, the term "laboratory research" refers to individual research conducted independently of evaluation campaigns [2]. This type of research often serves as a projection screen for critical analyses of the validity of scientific methods, as cited in many sources relevant to this thesis [2, 48–50, 65, 82, 98]. Additionally, it is usually smaller in scope compared to collaborative evaluation campaigns, backed by fewer resources, limited to reuse of fully published test collections and self-evaluated. Experiments in laboratory research often focus on measuring the relative improvement, i.e., effectiveness or efficiency, of a new retrieval approach compared to one or more baselines [44, 47]. All systems are thereby implemented on the same test collection with their performances evaluated by the same metrics and compared against each other. Any apparent improvement is then validated by statistical significance tests.

While such experiments are not fundamentally different from contributions to shared tasks, as both typically implement and test a novel retrieval system, the flexibility of the protocol for conducting experiments in laboratory research, combined with the self-evaluation of their results, makes it particularly vulnerable to a number of validity deficiencies [48–50, 59]. Against the backdrop of a generally less-than-ideal culture of scientific reality that favors "wins" and pressures with tight budgets and deadlines, research tends to focus on topics

that promise quick, good-looking results [67, 82]. Double checking on validity might become less of a priority [67, 82]. Free choice of baselines encourage the use of weak baselines, distorting actual progress [2, 62, 65, 98]. The lack of obligation in combination with limited resources in time often lead to the publication of non-reproducible results [66, 93]. Unrestricted access to fully published data prevents blind testing, which would reduce potential bias [77]. And the field's research focus on performance improvement instead of theory development prevents from conducting large-scale experiments, thus diminish tests for generalization [44, 47, 49]. The next section (2.2) discusses in detail these various shortcomings with respect to validity criteria in empirical science.

2.2 The Validity of Information Retrieval Experiments

Since research in information retrieval relies heavily on empirically obtained results, the validity of the underlying experiments is essential to ensure the validity of progress in the field itself. A common method for perceiving and qualifying validity in empirical research is to analyze it in terms of internal and external validity [44, 49, 50]. While internal validity refers to the cause-effect relationship within the setting of an experiment, i.e., whether the conclusions derived from the experiment are supported by the data, external validity assesses the extent to which these conclusions can be further generalized and transferred beyond their immediate context [44, 49, 50].

Reflections within the community about common research practices in information retrieval have highlighted a number of shortcomings related to this concept. The frequent use of weak baselines, as detected by several studies [2, 62, 65, 98], could bias measured effect sizes toward pseudoprogression [2]. This practice may be facilitated in part by a difficult-to-obtain overview of current state-of-the-art baselines [3, 65] that may be difficult to reproduce due to observations of a critical lack of reproducibility [1, 97]. Missing reproducibility, on the other hand, compromises the general validity of the results obtained [48, 66], which are almost always derived from fully disclosed data sets that are heavily reused. Such results are statistically questionable due to extreme value theory [20] and could be biased by data leakage from the outset [77]. External validity is often neglected in information retrieval experiments, since motivations have traditionally been strongly focused on performance optimization and investments into explaining their results through theoretical models are comparatively rare [47, 49, 50]. As a result, performance prediction in information retrieval is still nearly impossible [47]. Fuhr et al. strongly rec-

ommend that research should focus more on establishing theories by shifting the task of experiments to validate hypotheses rather then trying to improve performances [44, 47, 49, 50]. To this end they emphasize generalization testing through large-scale experiments as key to theory development, arguing that further progress lies in model prediction through theoretical foundations [47, 49, 50].

Awareness of these shortcomings has led to numerous approaches to address them. Several workshops of evaluation campaigns now promote research on reproducibility and generalization [42, 68]. In addition, the field of information retrieval research has seen continuous evolution in open-source software. The advent of computational methodologies like notebooks and containerization has significantly improved both the reproducibility and scalability of experiments. Processes that automate data processing and streamline complex procedures, which previously required substantial effort, can now be more easily achieved. This enables more efficient research and robust validation of results. With the advent of cloud computing, the Evaluation-as-a-Service paradigm has gained significant traction, providing a novel framework for conducting experiments and for addressing issues of internal and external validity [52, 53, 58]. This approach centralizes the evaluation process, ensuring standardized testing conditions, thereby promoting increased rigor and transparency in research, and creating opportunities for large-scale, collaborative, and comparative studies [52, 53, 58].

TIREx is built upon TIRA, a comprehensive Evaluation-as-a-Service framework that incorporates industry-standard technologies like Docker and Git. By integrating this platform with other contemporary research software, such as `ir_datasets` and PyTerrier, TIREx aims to further improve the validity of information retrieval research.

Baselines. To measure the improvement (or deterioration) of a retrieval approach, at least two systems have to be tested under the same conditions. For instance, to measure the effectiveness of BM25 compared to DirichletLM, both systems must be evaluated on the same data with respect to the same measure of precision. The distance between their scores of precision indicates the value of improvement. The system that is chosen as the reference point is called a baseline. Baselines are a foundation for empirical research in general [68] and for measures of improvement to be valid, in the broader context of progression, the right choice of baseline is vital [2, 62].

In 2009, Armstrong et al. questioned the supposed progress in research of ad-hoc retrieval over the past 10 years, due to the frequent use of weak baselines [2]. In a broad meta-study, they used the original results at TREC evaluation campaigns as baselines for all subsequent results of laboratory re-

search on the same collections of TREC Ad-hoc, Web, Terabyte and Robust reported at SIGIR (1998-2008) and CKIM (2004-2008). They found that most reported improvements diminished, when compared with the original TREC scores, often not even exceeding their median. A condensed timeline of reported improvements against the original TREC results over the past 10 years revealed not the expected slow but steady upward trend in improvement, but a persistent stagnation. They concluded that published results of improvements over incorrectly chosen baselines disguised the stagnation of actual progress. Kharazmi reaffirmed the ongoing usage of weak baselines [62], as did follow-up studies [65, 98].

There are various reasons for the use of weak baselines. Sometimes, access to cutting-edge approaches may be restricted due to commercial interests or proprietary technologies [62]. However, it is likely that strong baselines are much more often not readily available because they simply cannot be efficiently reproduced [65]. A study by Lin et al. highlighted the challenges of implementing strong, reproducible baselines from scratch [68]. Despite the potential of continuously improving research software and the upcoming of notebooks that address these issues, numerous obstacles to reproducibility persist. These obstacles can lead to inconsistent result scores for the same retrieval system [68, 97], or even break reproduction attempts entirely [64, 76]. However, this situation assumes that knowledge about a suitable baseline system exists in the first place. Awareness about state-of-the-art methods would benefit from centralized leaderboards [2, 49, 65] but their scarcity suggests failed attempts to establish such systems within the community [65]. Currently, information retrieval results are mostly scattered around the web, often buried in publications only, complicating efforts to gain a comprehensive overview of the actual research landscape.

The TIREx framework, which integrates the TIRA platform, is specifically designed to enhance both the reproducibility and replicability of retrieval pipelines in post-hoc experiments. By the initial implementation of 50 retrieval systems across 31 test collections, it already offers a broad range of robust baselines. Furthermore, these baselines are aggregated in publicly accessible leaderboards[6], which serve as resources for tracking progress and establishing reasonable baselines in the field.

Leaderboards. Multiple studies over the last 15 years reported about the regular use of too weak baselines in information retrieval experiments, publishing noisy improvements and thereby distorting actual progress [2, 62, 65, 98]. One suggestion to improve on such practice would be to promote central leader-

[6] `https://www.tira.io/task/ir-benchmarks`

boards to the community [2, 49, 65].

While leaderboards may encourage part of research to spiral into SOTA-chasing[7], with its ambiguous effects such as fostering competition over collaboration and diverting resources from more significant and enduring research objectives [22], the benefits of centralized leaderboards may prevail such tendencies. Frequently popularized and well maintained, they may help to simplify deep analysis of effectiveness development [2], which could be used in building theoretical models [47]. Additionally, they would inform about contemporary baselines, which may potentially raise community standards [2, 65]. Despite these benefits, all past efforts to establish such centralized general-purpose leaderboards seem to have failed. For instance, following their study highlighting the issue of weak baselining, Armstrong et al. launched EvaluatIR, a general leaderboard that accepted run file submissions [3]. This endeavor proved to be unsuccessful from the outset, and similar initiatives never gained momentum either [65]. Task- or collection-scoped leaderboards seem to be more readily adopted, with the MS MARCO leaderboard[8] being the most prominent, accepting run files submissions. However, proposals have been made to enhance the conceptual implementation of leaderboards [70]. Publicly accessible submissions, especially run files, may leak information about the test data, thus undermining the validity of the test collection with each submission over time [70]. To address this issue, Lin et al. propose an Algorithm-to-Data approach [53] in which the leaderboard accepts software rather than run file submissions. An infrastructure supporting the leaderboard would execute this software on a blind held-out test, thereby preventing data leakage and archiving the software for future reproduction [65, 70]. To further counter non-innovative approaches, such as plain ensembles of the leading top submissions, best practice would also provide that public access to submitted software be restricted, at least for a period of time [70].

By adhering to the Algorithm-to-Data approach and enabling blind testing along with administration of access to submissions and software, TIREx successfully implements the recommendations made by Lin et al.. As an infrastructure designed to host and centralize experiments from small-scale laboratory research to large-scale shared tasks across various test collections, TIREx further presents a promising solution for a general leaderboard.

Reproducibility. Ensuring the reproducibility of scientific experiments is crucial for maintaining their internal validity [47, 48]. Results that cannot be reproduced may have simply happened by chance, thereby undermining their reliability [47]. Understanding science as an iterative process of knowledge

[7]SOTA: state-of-the-art

[8]https://microsoft.github.io/msmarco/

accumulation, where new research builds upon previous findings, reproducibility serves as a cornerstone that accelerates scientific progress. It minimizes potential dead ends and reduces waste of investments [43, 67].

In this thesis, the term "reproducibility" refers to the process where researchers use a third-party artifact to confirm results that were initially repeatable. "Repeatable", in this context, means that the original developers could reliably reproduce the results using their own artifact [67].

In computational sciences, processes are inherently deterministic, intuitively suggesting intrinsic repeatability with the idea that executing an algorithm should, in theory, consistently yield the same results. In the field of information retrieval, the Cranfield paradigm further bolsters this notion of inherent reproducibility, given that the data upon which the algorithms operate is also fixed [43]. However, various investigations into this matter had proven otherwise [31, 43, 93]. The growing awareness of this has been reflected in several initiatives to promote reproducibility at information retrieval conferences, such as dedicated reproducibility tracks, OSIRRC[9] [68], CENTRE lab at CLEF, TREC and NTCIR, and the ACM SIGIR Artifact Badging Board[10] [42], which recognizes successfully reproducible publications, among others. Continuous advances in community-developed research software, such as comprehensive search engines, further enhance reproducibility. Additionally, the advent of computational notebooks, specifically designed to facilitate the sharing of computational processes, inherently promotes reproducibility. However, a large study into the reproducibility of Jupyter notebooks[11] demonstrated that reproducibility is not yet a foregone conclusion: out of 850,000 notebooks, only 24% executed without error and just 4% reproduced the reported results [76]. These numbers do not differ significantly from results of a previous study, which investigated the reproducibility of source code not encapsulated in notebooks [31]. This suggesting that many challenges in achieving reproducibility still remain unresolved. Part of the problems are due to human factors. Standard source code is often coupled with incomplete documentation, which becomes more difficult through hidden states and the escalating complexity of retrieval pipelines [43, 64, 93]. Notebooks often suffer from poor coding habits that can hinder reproducibility. Common issues include code breaking execution orders, as well as poor practices in naming, versioning, and modularization [76]. From a technical standpoint, a significant problem shared by both, code repositories and notebooks, is the lack of data accessibility and unresolved external dependencies [64, 67, 93]. Given the constant evolution of hardware and software, maintaining reproducibility becomes an ongoing process in itself [67].

[9] The Open-Source IR Replicability Challenge: `https://github.com/osirrc`

[10] `https://sigir.org/general-information/acm-sigir-artifact-badging/`

[11] `https://jupyter.org/`

Hanbury promotes the Evaluation-as-a-Service framework as a solution to improve reproducibility [53], since it offers a uniform environment for running experiments with the ability to archive code and data, providing a means for continuous verification. TIREx integration with TIRA supports continuous integration, which archives experiments together with its results, logs, data, metadata and software in automatically generated git repositories. The support of Docker for software submissions, executed in environments without internet access on read-only mounted data, further bypasses the problem of external dependencies, as they are already loaded before snapshotted.

Blinded Experiments. In scientific experiments, information is often concealed from participants to minimize experimental biases, namely observer bias or confirmation bias[12]. Although blinded experiments are commonplace in many scientific disciplines, such as biomedical science, physics, or social science, this practice has not been widely adopted in the field of information retrieval where data is typically accessible to all participants [77]. Evaluation campaigns that incorporate shared tasks are somewhat semi-blinded since relevance judgments are generally not provided in advance [43]. However, these tasks primarily adopt a Data-to-Algorithm approach, where organizers distribute the documents and topics for implementation, thus unblinding them for the participants [53]. Laboratory research is usually completely limited to experiment on fully published data, given the resource constraints that prevent the creation of new test collections from scratch. Furthermore, the necessity for self-evaluation grants researchers access to relevance judgments.

Evaluation-as-a-Service platforms promote an Algorithm-to-Data approach where software submissions replace run files [53]. This eleminates the need to distribute any data and enables the execution of blinded experiments [77]. In the context of a new test collection, one can envision it remaining blinded, safeguarded by the structure of such platforms (albeit tied to them). It would be even possible to collect information about the number of experiments performed on it to account for extreme value theory statistics [20]. As TIREx is integrated with TIRA, it fully supports blinded experiments.

Large-Scale Experiments and Tests for Generalization. The degree of external validity is measured by the extent to which results, derived from a particular focus, can be extrapolated to a broader context [44, 49]. Consequently, to ascertain generalization, it is essential to be able to make informed predictions about outcomes based on some sort of theoretical model [44, 47].

Since research in information retrieval predominantly concentrates on model

[12] `https://en.wikipedia.org/wiki/Blinded_experiment`

improvement [44, 47], predicting model behavior remains challenging due to the scarcity of theoretical models [49]. Constructing these models necessitates a research perspective that seeks the reasons for the success of a phenomenon rather than the means of its achievement [44, 47]. In empirical research, this translates to the acquisition of extensive empirical evidence via large-scale experiments conducted across varied data [44, 47]. Fuhr et al. emphasise the importance of performing such studies. They promote iterating experiments across different collections and diverse tasks to amass adequate data that could form the groundwork for theoretical hypotheses [44, 47].

Hanbury et al. suggest employing the Evaluation-as-a-Service framework to address this challenge, as its features would facilitate scalability of experiments and centralize result accumulation over time [53]. This structure would be further enhanced if experiments could readily be rerun on different data sets or repeated with different retrieval software.

The TIREx framework is specifically designed to enable modular retrieval pipelines. It standardizes the input and output for indexing, retrieval, and evaluation tasks, which allows for interchangeability of data sets and software. As a result, the replication of experiments on different test collections or with different retrieval software becomes almost effortless, which promotes research towards generalization by large-scale experiments.

2.3 Evaluation-as-a-Service

Experiments in information retrieval are typically resource-intensive. Modern test collections regularly exceed sizes of over 25TB in uncompressed data [97], and upcoming collections are likely to be even larger. This trend not only makes these collections challenging to distribute and implement, but also means that operations on such vast data are time-consuming and reliant on high computational power [53, 97]. Additionally, new tasks in information retrieval, such as retrieving confidential data or working with real-time data, require novel experimental setups, given that these data sets cannot be distributed [52, 53]. The advent of cloud computing has allowed for the development of structures that separate research groups from the actual data and computation by moving the execution and evaluation of information retrieval experiments to remote servers [52, 53, 58, 82].
Typically, experiments in information retrieval follow a Data-to-Algorithm approach, where the researchers possess the test collection and implement software onto it [52, 53, 58]. Evaluation-as-a-Service platforms (EaaS), however, implement an Algorithm-to-Data approach, where the software is submitted

to run on an infrastructure that houses both the data and the computational environment [52, 53, 58]. Beyond its ability to make non-distributable data accessible for testing, this approach also alleviates technical burdens on research groups. This can level the playing field among participants in shared tasks [53]. Furthermore, the framework has the potential to improve both the internal and external validity of information retrieval experiments. By withholding data from participants, it becomes possible to conduct blinded experiments, which can prevent data leakage and bias from influencing results [77]. An infrastructure that archives both submitted software and its dependent data in a stable environment improves reproducibility. This improvement applies not just to individual experiments but potentially to entire shared tasks. Over time, aggregating data across different test collections and tasks also provides insights for generalization. [53].

Given the scope of this thesis, I will only present the TIRA platform, which is the EaaS of choice to implement TIREx. For a detailed overview of EaaS platforms, refer to Hopfengartner et al. [58].

TIRA. **T**he **I**ntegrated **R**esearch **A**rchitecture (TIRA) [51, 77], initially introduced in 2012, has been utilized for organizing shared tasks involving software submissions, including PAN and Touché, hosted at CLEF. The original version of TIRA allowed participants to access virtual machines for software submissions during shared tasks. However, this approach proved to be non-scalable and error-prone, making it challenging for external researchers to reproduce the submitted software. As a result, TIRA underwent a complete redevelopment based around industry-standard continuous integration and deployment (CI/CD) pipelines, implementing Git, Docker, and Kubernetes into its framework [46]. In the new version of TIRA, participants upload their software as Docker images to a private Docker registry dedicated to their team, blinding their software for other teams until unblinding them is administered. For on-demand execution, TIRA runs the software in a Kubernetes cluster with significant computing resources. Software execution and evaluation are each sandboxed (i.e., cut off from the internet), preventing data from leaking, as well as the download of external dependencies. This ensures truly blinded experiments and improving reproducibility. Both retrieval software and evaluation are snapshotted and merged automatically into the tasks git repository with all necessary metadata if execution was successful. This updated version of TIRA was first employed in large-scale NLP tasks at SemEval 2023, with 170 registered teams and 71 teams submitting results. However, when setting up the retrieval-oriented Touché task for CLEF 2023, it became apparent that TIRA still had limitations for information retrieval tasks. These shortcomings included the lack of unified data access, inefficient implementation of typical

information retrieval workflows, the inability to separate between full-rank and re-rank software and the ability to modularize retrieval pipelines into components with caching. This issues were addressed by integrating `ir_datasets` [71], PyTerrier [72] and `ir_measures`[13], to which this work contributed. Details about these implementation are provided in Chapter 3.

2.4 Open Source Research Software in Information Retrieval

The field of information retrieval has witnessed significant advancements through the contribution of open-source software, providing comprehensive frameworks designed specifically for information retrieval tasks. Some of these frameworks have evolved into fully-functional search libraries, covering a wide range of functionalities and facilitate various aspects of the retrieval process, including the implementation of end-to-end retrieval pipelines from indexing, to query processing, to document retrieval. Notably, Apache Lucene[14] and the Lemur project[15], both launched in the early 2000s, have emerged as pioneering and influential examples of these libraries. Various search engines like Anserini [96], Galago [21], Indri [83] and Terrier [74], among others, are partly build upon the foundations laid by these libraries, further enhance them with additional features tailored specifically for information retrieval research. These features encompass the implementation of prominent retrieval algorithms and the support for performance evaluation, streamlining the execution of information retrieval experiments. This allows researchers to focus more on their specific research goals by significantly simplify the implementation of complex techniques and promote the reproducibility of their research. [41, 97]. Additionally, a wide range of open-source libraries and toolkits have been developed, which specialize on certain tasks, namely facilitate the access to test collections and evaluation protocols, document indexing, topic modeling and many more. With this work, we integrated `ir_datasets`, PyTerrier and `ir_measures` into TIRA.

`ir_datasets` [71] is a Python library, developed by the Information Retrieval Lab at Georgetown University to simplify data handling. It provides streamlined access to a wide range of text based test collections commonly utilized in information retrieval research. Its standardized API presents a consistent interface for handling diverse data sets, regardless of their original format or

[13] `https://github.com/terrierteam/ir_measures`
[14] `https://lucene.apache.org/`
[15] `https://www.lemurproject.org/`

structure. In addition to the core text data, the library facilitates access to various types of metadata and auxiliary information. It support caching for faster repeated data loading and provides iterable data loading functions to accommodate large data sets. We integrated `ir_datasets` into TIRA to standardize the input of documents and topics for indexing, the relevance judgments (qrels) for evaluation and the input of topics and qrels for re-ranking purposes.

PyTerrier [72] is an open-source Python toolkit, built on top of the Terrier search library, providing a pythonic interface for information retrieval processes with a strong focus on experimentation and performance evaluation. It supports a wide range of retrieval systems, from prominent term-weighting models to neural ranking approaches and facilitates the implementation of complex retrieval pipelines. PyTerrier integrates seamlessly with popular Python data science libraries like Pandas and scikit-learn, enabling sophisticated data manipulation and machine learning workflows within the experimentation process. Furthermore, PyTerrier's integration with the `ir_datasets` library supports data loading from a wide range of standard information retrieval data sets. We use PyTerrier to index the test collections and for conducting most of our retrieval processes with its integrated retrieval models.

`ir_measures`[16] is an open-source Python library that provides a comprehensive interface for evaluating information retrieval systems. It acts as a bridge between various integrated information retrieval evaluation tools, such as cwl_eval [4], pytrec_eval [85] and trectools [75], offering a unified API to access a broad range of evaluation metrics. The library supports more then 30 measures and diversity metrics. Its compatibility with different judgment types, including binary and graded relevance, caters to various evaluation scenarios and in addition to single-query evaluations, `ir_measures` supports query-level and system-level aggregations. We use `ir_measures` to measure the retrieval processes conducted within TIREx.

[16] `https://github.com/terrierteam/ir_measures`

Chapter 3

The Information Retrieval Experiment Platform

The **I**nformation **R**etrieval **Ex**periment Platform (TIREx) [45] is an assemble of contemporary information retrieval research software integrated with the TIRA Evaluation-as-a-Service framework, forming a platform that accommodates, administers, and executes information retrieval experiments, ranging from individual lab studies to shared tasks. Its integration with TIRA enables software submissions, provides administrative features for selective data blinding, and supports self-containing Git archiving of experiments for post-hoc reevaluation. It thereby enhances internal validity, including reproducibility. Its architecture compartmentalizes all phases of the retrieval pipeline through a unified input/output system (i.e., data set loading, indexing, retrieval and evaluation), achieved by the integration of a wrapped `ir_datasets` instance, which permits for rapid modifications to pipeline components and enables caching features to enhance performance. This design significantly facilitates large-scale experiments, thereby boosting external validity by encouraging research into generalization.

In the following, I first deliver a brief summery of how TIREx works. Next, I elaborate on how we integrate `ir_datasets` with TIRA to standardize the access to a large amount of common test collections. I further describe how we also use it to unify the data flow between software components in sophisticated information retrieval pipelines through task definition, effectively making components exchangeable. Subsequently I provide examples of how common retrieval pipelines with TIREx are structured and elaborate on the evaluation processes with the `ir_measures_evaluator`. I conclude with an exploration of the effective remote reuse of experiments in standalone PyTerrier pipelines. This is facilitated through the self-containment of TIRA's CI/CD Git archives and the TIRA API, boosted by its caching capabilities. These developments

lay the groundwork for the next chapter, where we extensively leveraged these features to conduct three large-scale experiments.

3.1 Conducting Experiments on TIREx

Figure 3.1 presents an abstract sequence diagram that outlines the principal process of a common retrieval experiment on TIREx. TIREx loads data sets into TIRA in a standardized way through a Docker image of the `ir_datasets_loader`, a wrapper for the `ir_datasets` library. Users can select from already integrated (unblinded) collections. New data sets can be integrated by adding them to the `ir_datasets` package, which is open to pull requests and supports the administration of private access, making it suitable for confidential data. Nevertheless, TIREx also accommodates the import of data from individually defined sources to cater to specific use cases, such as the requirement for live data or for strict confidentiality. Loaded data sets are indexed by a dockerized indexing software and cached for future use. Retrieval approaches are submitted to TIRA as software in Docker images. Any additional input data (e.g., a manually re-ranked run) is submitted via run uploads. For execution, TIRA copies the submitted software to a sandboxed environment, linking it to the index. This environment isolates the running software from the internet, preventing potential data leakage and enhancing reproducibility by ensuring the software is fully installed beforehand. Subsequent evaluation is also executed in a sandbox, with the run file evaluated by a dockerized instance of the `ir_measures_evaluator`, which incorporates `ir_measures`. Successfully conducted experiments are automatically versioned and archived via CI/CD pipelines in dedicated git repositories, including all software, run uploads, results, logs, and metadata, ready for publication if administered. These archives are self-contained, and the unified data processing by our implementation of `ir_datasets` allows for further independent execution in PyTerrier pipelines, even extended to different data sets. This feature elevates all information retrieval experiments conducted on TIREx to be potential participants in other experiments, including large-scale investigations.

3.2 Unifying Input Data with `ir_datasets`

The `ir_datasets` module enables unified access to text based test collections commonly used in information retrieval research and is embedded as a data layer in a multitude of IR research software including PyTerrier.

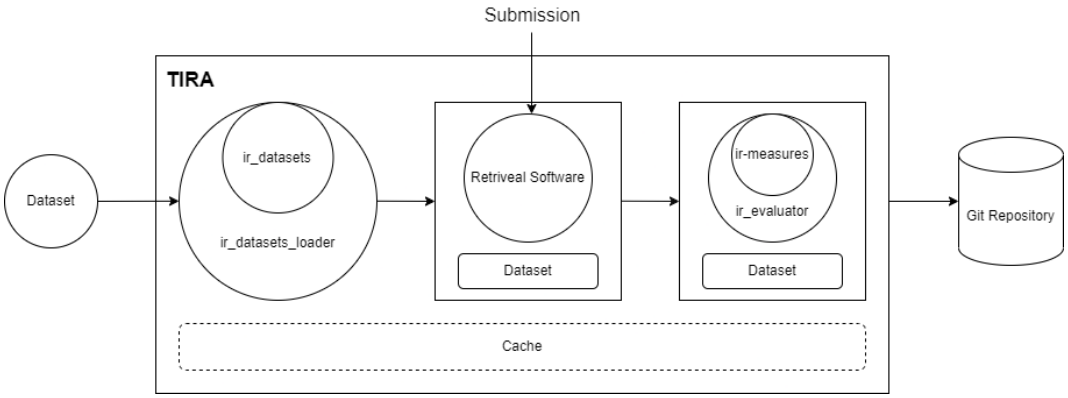

Figure 3.1: The TIREx experiment pipeline.

With TIREx, we implement `ir_datasets` within a custom wrapper into TIRA to process test collections for retrieval experiments in a standardized manner. This integration is compartmentalized from PyTerrier, yet fully compatible with PyTerrier pipelines. With our implementation, retrieval systems can access all structured information if selected, while we extend it with a "default text" field for documents and topics to ensure a uniformly defined interface between various data sets and retrieval software. In addition, we provide the option to select between full-ranking and re-ranking tasks. For the former, entire corpora are imported into TIRA, while for the latter, the documents and topics of a corpus are loaded, to create re-ranking files for any specified run files. The integration of `ir_datasets` into TIRA is achieved using Docker images of the `ir_datasets_loader` with their configuration defined by the to-be-executed command of the Docker container, in accordance with the command-line interface of the `ir_datasets_loader`.

The `ir_datasets_loader` serves as a transformation layer for the `ir_datasets` package and provides standardized data formats for all data sets accessible via `ir_datasets` that incorporates the "default text" field. It distinguishes between data creation for full-rank and re-rank operations and ensures full compatibility with PyTerrier. For full-rank operations, the `ir_datasets_loader` maps documents, topics, and relevance judgments from the chosen data set into a unified structure. In re-rank mode, it extracts documents and topics based on a given run file, aggregating the data to create re-ranking files suitable for re-ranking operations. Additionally, the system provides the possibility to include or exclude the entire structured information of documents and topics in the output. This supplements the data that has been assessed as the main content and is accessed via the newly introduced default text method. A command-line interface with five primary arguments enables the configuration

of the ouput of the `ir_datasets_loader`, as detailed below:

- `ir_datasets_id`: This is the `ir_datasets` identifier for the specific data set to be loaded, provided as a string.
- `output_dataset_path`: Specifies the path where the output will be saved, represented as a Python `Path` object.
- `output_dataset_truth_path`: An optional argument designating a separated path for the relevance judgments to be saved, represented as a Python `Path` object.
- `include_original`: A boolean value that indicates whether the entire document with all structured information should be stored within the output.
- `rerank`: string 'rerank' or None, this argument activates the re-rank operation of the `ir_datasets_loader`, generating files specifically tailored for re-rank operations.

In the following, I detail our approach to integrate various data sets and retrieval software through two key implementations: First, I describe our introduction of a unified "default text" field into `ir_datasets`, streamlining data access across all integrated test collections. This facilitates the interchangeability of both collections and retrieval software. Subsequently, I elaborate on the preservation of exchangeability and concatenability of arbitrary retrieval software in modular retrieval pipelines through the differentiation of tasks by the `ir_datasets_loader`.

Data access unification with default text fields. While `ir_datasets` wraps test collections into unified data types (i.e., namedtuples and distinct class names for documents, topics and relevance judgments) test collections still vary in the way they structure their information internally. While some corpora provide only a single text field for each document and topic, others enrich them with additional information and metadata. For instance, the MS MARCO passage corpus [39, 40, 73] provides only a single text field for each document and topic. Conversely, the ClueWeb TREC-Web 2009-2014 corpora [25–28, 32, 33] store additional data, including URL, date, HTTP headers, and content type for each document. In addition, each topic within these corpora is supplemented with a description, type, and nested subtopics field. Furthermore, the keys used to access these fields can vary. The MS MARCO passage corpus key for the main text field of both, documents and topics, is just "text". However, in the ClueWeb TREC-Web corpora, the key for the main text field in documents

is "body", while the keys to access the topics are either "query" for the user-like query-string or "description" for a more detailed version.

Rich information can reflect special use cases and provide additional data to support tailored retrieval systems that seek to improve document retrieval by processing specific data structures. For instance, a retrieval model might filter documents by content type prior to document retrieval or utilize a query description to elaborate on brief input queries. These retrieval systems necessitate adaptations to process the specific supplemental data. If a retrieval system ignores this additional data, the focus then shifts to its general search capabilities and its adaptability to diverse search contexts.

In order to conduct large-scale experiments to investigate generalization properties, a common experimental framework for all subsidiary experiments is essential. For instance, if a retrieval system is to be investigated for its general search qualities, it should be evaluated across numerous test collections under conditions that are as uniform as possible. Additionally, to facilitate such experiments, the system's implementation across different test collections should be as straightforward as possible.

To accommodate these scenarios and to boost research into generalization, we introduce a `default_text` field for each document and each topic within `ir_datasets`. This feature is directly integrated into the `ir_datasets` package and allows to access the main content of each document and topic by the unifying `default_text` method of the document and topic classes, processing the content as a single string. Assessment for a query or document's default text field needs to be manually reviewed but is often straightforward. For instance, Medline documents consist of a title and an abstract, in which case the default text would be a combination of both fields. Numerous test collections comprise title, description, and narrative versions for topics. As the title typically pertains to short queries that users would input into search engines, we utilize the title as the default text in most cases. In instances where the choice is not as obvious, we strive to authentically represent the document or topic's most crucial content, while also being open to suggestions from the community via pull requests.

Flexible retrieval pipelines through task differentiation. In modern retrieval approaches, a retrieval pipeline often consists of one retrieval system re-ranking the ranking from a prior retrieval process. This operation involves at least two stages with the primary stage performing an initial full-ranking procedure of a document corpus corresponding to a set of queries, which is then condensed into a run file. In a subsequent stage, a re-ranking operation is performed on the ranking previously recorded in that run file. Therefore, a minimum of two retrieval systems is necessary. The full-ranker accepts the documents and

topics of a test collection as input and produces the run file as output. On the other hand, the re-ranker uses a specific re-ranking file derived from the content of the prior outputted run file as input. This file consists of document-query pairs along with the score and rank for each entry of the run file. It then yields another run file as output. Retrieval operations like these can be escalated to several stages, linking multiple re-ranking processes sequentially, initially triggered by a full-ranking retrieval operation at the outset.

With TIREx we aim to facilitate modular retrieval pipelines where retrieval software can be easily chained and exchanged without adopting the respective retrieval software. We use the `ir_datasets_loader` as a bridge in between two retrieval operations, collecting all additional data needed via `ir_datasets` to automatically building re-ranking files from run files, if necessary. As a result, re-rankers can operate on any previous stages that outputs run files in TREC format, as the output of these stages is converted into a standardized format.

Figure 3.2 and 3.3 provide diagrams about the general input/output conversion of the `ir_datasets_loader`, while listing 3.1 and 3.2 give exemplified insight into the data structure that the `ir_datasets_loader` produces for full-ranking and re-ranking operations. For full-rank software the `ir_datasets_loader` generates a `documents.jsonl.gz` file, which comprises the document ID (`docno`), the default text field (`text`), and the full document structure with all its individual fields accessible through the `original_document` field if this option was set at execution. It further generates a `topics.jsonl.gz` file with the query ID (`qid`), the default text field (`query`) for the main query data and the whole topic structure with all its fields stored in the `original_query` field. Both files are in JSON Lines format. Additionally, the topic file structure is furthermore mapped and exported to a trecxml formatted file.

For re-rankers, the `ir_datasets_loader` creates a `re-rank.jsonl.gz` file in JSON line format using the run file from the preceding stage. The re-rank file includes entries of query-document pairs to be re-ranked, along with the score and rank given by the prior stage. Re-rankers are then tasked to reorder all documents into their output run file. In this case, the `ir_datasets_loader` is run prior to the re-ranker, with the re-ranker only receiving the `re-rank.jsonl.gz` file as input.

If the data set contains relevance judgments, the `ir_datasets_loader` further provides a `qrels.txt` file for both tasks in standard TREC format.

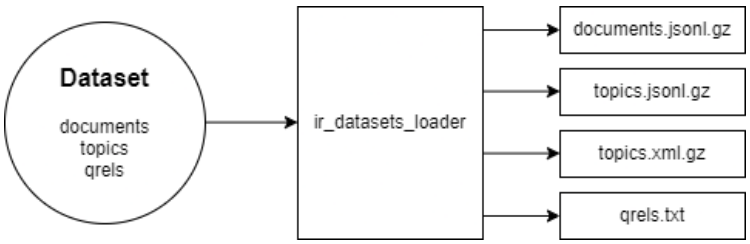

Figure 3.2: I/O of the `ir_datasets_loader` for full-ranking operations.

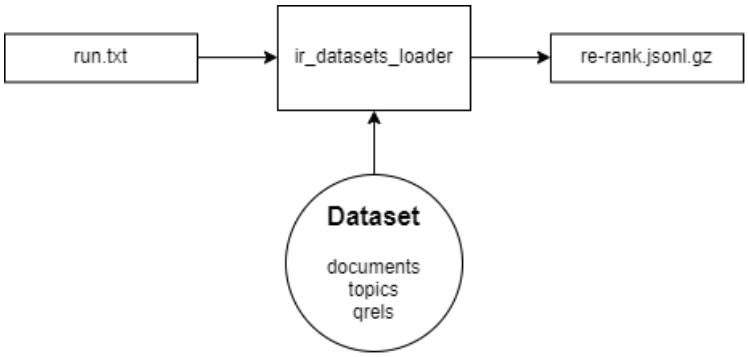

Figure 3.3: I/O of the `ir_datasets_loader` for re-ranking operations.

3.3 Running Retrieval Pipelines with TIREx

TIREx utilizes the TIRA platform to run and archive information retrieval experiments. TIRA uses Docker images for software implementation, which promotes the modularization of retrieval pipelines. The immutability principle that TIRA applies to submitted software guarantees its repeatability, allowing for the reliable caching of outputs. The cached output can be shared across all pipelines, significantly increases overall efficiency.

Our implementation of `ir_datasets` within the `ir_datasets_loader` unifies the integration of test collections for the assembly of retrieval systems and provides for the differentiation of retrieval software into full-rank and re-rank systems. The standardized processing of output from retrieval systems to input for rerankers, combined with TIRA's caching capabilities, make retrieval pipelines in TIREx easily scalable and their output can be reused by other pipelines.

Data management within TIRA. To manage the flow of data between the software components of retrieval pipelines, TIRA uses up to three variables that reference data stores passed to each software. These variables can be used in the to-be-executed commands of Docker containers and are also globally available.

Listing 3.1: `Example entries:` `documents.jsonl.gz` (left), `topics.jsonl.gz` (right)

```
{
    "docno": "MED-48",
    "text": "Agricultural policies,
         food and public health
        Abstract A historical view
        on how our agricultural
        systems evolved and how
        they are contributing to
        obesity and disease.",
    "original_document": {
        "doc_id": "MED-48",
        "url": "http://www.ncbi.nlm
            .nih.gov/pubmed
            /21151043",
        "title": "Agricultural
            policies, food and
            public health",
        "abstract": "Abstract A
            historical view on how
            our agricultural
            systems evolved and how
             they are contributing
            to obesity and disease.
            "
    }
}
```

```
{
    "qid": "PLAIN-1017",
    "query": "detoxification",
    "original_query": {
        "query_id": "PLAIN-1017",
        "title": "detoxification",
        "all": "detoxification - -
            cancer, raw food, heart
             health, heart disease,
             industrial toxins,
            women 's health,
            pregnancy, persistent
            organic pollutants,
            meat, infants,
            cardiovascular health,
            cardiovascular disease,
             complementary medicine
            , fish, alternative
            medicine - -"
    }
}
```

The variable `$inputDataset` points to a directory containing all files of the input data set passed to a software. For instance, for a full-ranking retrieval pipeline, evaluated on a specified test collection, the directory would hold the `documents.jsonl.gz`, `topics.jsonl.gz` and `qrels.txt` files of that specific test collection, serving the first two to the indexing software and the last two to the evaluator.

The variable `$inputRun` is only available for retrieval software composed of multiple components and points to a directory with the output data of all preceding components. For instance, in a re-ranking pipeline, consisting of one full-ranker and two re-rankers, the output of all run files from each software would be collected in the directory referenced by `$inputRun` in the order in which they were defined. The `ir_datasets_loader` that builds the `re-rank.jsonl.gz` from each run file, as well as all subsequent evaluators, would obtain the run files via this variable.

The variable `$outputDataset` points to the directory at which TIRA expects the output of each software, before it is redistributed (e.g. to the directory referenced by `$inputRun`).

Listing 3.2: Example entry of `re-rank.jsonl.gz`

```
{
    "qid": "PLAIN-1017",
    "query": "detoxification",
    "original_query": {
        "query_id": "PLAIN-1017",
        "title": "detoxification",
        "all": "detoxification - - cancer, raw food, heart health,
            heart disease, industrial toxins, women 's health,
            pregnancy, persistent organic pollutants, meat, infants,
            cardiovascular health, cardiovascular disease,
            complementary medicine, fish, alternative medicine - -"
    },
    "docno": "MED-48",
    "text": "Agricultural policies, food and public health Abstract A
        historical view on how our agricultural systems evolved and
        how they are contributing to obesity and disease.",
    "original_document": {
        "doc_id": "MED-48",
        "url": "http://www.ncbi.nlm.nih.gov/pubmed/21151043",
        "title": "Agricultural policies, food and public health",
        "abstract": "Abstract A historical view on how our
            agricultural systems evolved and how they are contributing
             to obesity and disease."
    },
    "rank": 1,
    "score": 45.67
}
```

Manual components as input. Some retrieval processes may use data specifically crafted for that very process, so called manual submissions at evaluation campaigns. For instance, the Interactive Track of TREC, which ran from 1996 until 2003 [6–11, 56, 81], often involved manual aspects, as it focused specifically on user interaction with search systems. Thus, a user might for example perform an interactive search process and then manually select documents to be included in the final result set [56]. Use cases like these are supported by manual uploads. Uploaded files can be configured as preceding components and therefore made available as input to subsequent components via the `$inputRun` variable.

Improved efficiency through caching. All submitted software to TIRA is immutable, hence the output of each software is considered reliable. This feature allows caching of outputs, which enables their reuse in new contexts. Once computed, outputs can be shared if appropriately managed. Retrieval pipelines thus gain efficiency by circumventing the need to recompute outputs from previously executed software, since cached outputs can be directly utilized. For instance, a pipeline may solely calculate a final re-ranking, ignoring preceding stages if the same software has processed these stages once

before. Since software reruns remain possible, multiple instances of the same output may exist. In such cases, the output from the first execution serves as the cache for future reuse. Moreover, output that has served as input for other software cannot be deleted as long as dependent software exists. On the other hand, software can only be deleted if all of its outputs are deleted first. Any programming errors can only be fixed by submitting new software. Thus, TIRA's immutability principle enhances system efficiency, maintains integrity, and assures repeatable and dependable outputs.

Examples for retrieval pipelines with TIREx. Common retrieval experiments carried out with TIREx usually involve full ranking or re-ranking pipelines. These processes commence with the loading of a test collection, followed by its indexing, subsequent retrieval operations, and its evaluation.

Figure 3.4 illustrates a full-ranking approach, demonstrating the data flow both with and without the presence of corresponding caches. All output of each software is cached and if a cache of a software's output is available, subsequent stages would be served from the cache instead. In the absence of caches, the Docker image containing the `ir_datasets_loader` is executed with the specific arguments for a given data set and task and all output is stored in `$outputDataset`. These are made accessible by TIRA in `$inputDataset`. Subsequently, the indexing software builds an index within `$outputDataset`. TIRA then avails the index from `$outputDataset` to the full-ranking software within `$inputRun`, channeling its output back via `$outputDataset` into `$inputRun`. Finally, the evaluator software uses the run file from `$inputRun` and the data set from `$inputDataset` to assess the retrieval process, recording its evaluation back inside `$outputDataset`.

Figure 3.5 highlights details of a more complex re-ranking process from a retrieval pipeline, featuring multiple chained re-rankers subsequent to a full-ranker, including a manual submission. The full-ranker outputs a run file to `$outputDataset` which TIRA then makes accessible within `$inputRun`. Subsequently, the `ir_datasets_loader` builds a `re-rank.jsonl.gz` file from the run file and outputs it to `$outputDataset`, from where TIRA avails it within `$inputRun`. For the first re-ranker a manual submission (e.g., manual query formulations) is served via upload to `$inputRun`. The re-ranker accesses all files from this location, directing its run back to `$outputDataset`. TIRA then gathers the data and serving all subsequent stages after each process, aggregating all output inside `$inputRun`.

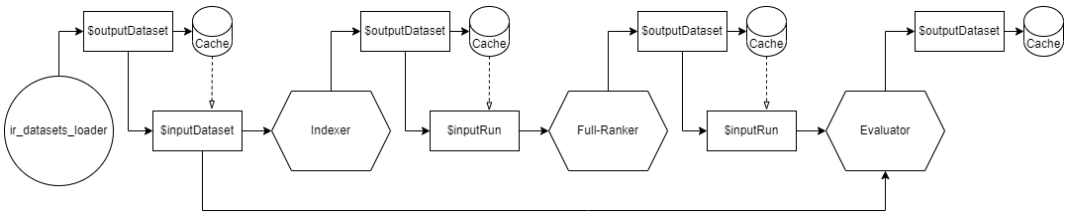

Figure 3.4: Example of a full-ranking pipeline on TIREx.

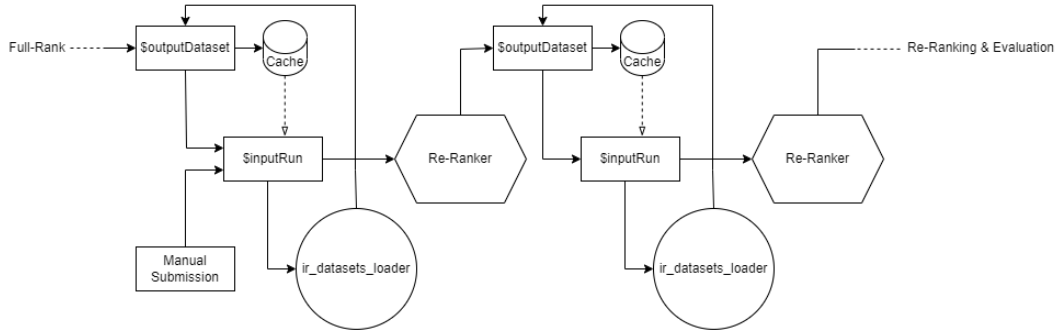

Figure 3.5: Excerpt from a re-ranking pipeline with a manual submission on TIREx.

3.4 Evaluation with the `ir_measures_evaluator`

To evaluate retrieval processes within TIREx, we integrated a comprehensive default evaluator via Docker image into TIRA that utilizes `ir_measures`.

Evaluations are performed in a sandbox environment separate from the Internet to ensure data privacy, and are automatically processed each time a retrieval pipeline is executed. Without topics and relevance judgments passed, the evaluator checks the run file for its integrity, providing comprehensive feedback about any perceived issues. It thereby distinguishes between warnings and errors that would break any further evaluation process. In addition to several I/O inspections, it checks for the correct number of columns, multiple tags or special characters within tags, special characters within query identifiers, their correct ascending order, special characters within document identifiers, unexpected values in the ignored column, if scores are floating point numbers, the correct ascending order of scores, the appearance of score ties, if ranks are integers, the correct ascending order of ranks and if ranks are consecutive and consistent in their order.

When topics and relevance judgments are passed, the evaluator checks the run file and, assuming it can be parsed, checks its consistency regarding its

Listing 3.3: Code snippet for a full-rank retrieval from a complete corpus.

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

Listing 3.4: Code snippet to run a re-ranker from an experiment.

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

document and topic identifiers against the topics and the relevance judgments file. It also checks the consistency of the topic and relevance judgments file regarding not shared identifiers. If metrics are passed, it runs all checks and then evaluates the run file against the metrics. The evaluation is computed as an average across all queries and individually for each query to allow for later statistical analysis. The evaluation results are then exported in prototext format.

3.5 Reusing Experiments in PyTerrier Pipelines

Experiments successfully run on TIRA are automatically versioned and archived in dedicated git repositories. This includes all results, such as run files and evaluations, as well as software submissions in the form of Dockerfiles, with the images loaded to Dockerhub. All experiment repositories are self-contained and work stand-alone, independent from TIRA, with privacy settings available. If public access is permitted, they become remotely accessible through the TIRA API for post-hoc reproducibilty and replicability studies. Through our standardization efforts, artifacts of experiments conducted on TIREx are reliable to work seamlessly with `ir_datasets` and in PyTerrier pipelines and can be reused in further experiments, reaching out to large-scale experiments for generalization tests. Each software component of an experiment is thereby identified by `<task-name>/<user-name>/<software>`, whereas `<task-name>` refers to the specific (shared-)task, `<user-name>` to the team that conducted the experiment and `<software>` to the submitted software.

Listing 3.3 shows how full-ranking software can be repeatedly executed on the original data set or even rerun on arbitrary data sets and subsequently used as preceding components in advanced PyTerrier pipelines. The `retriever`

Listing 3.5: Code snippet to re-rank a run by a submission software.

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

method uses the software identifier and a data set as arguments to execute the software on the data set by downloading and executing the referenced Docker image. Any data set available in `ir_datasets` is suitable. This retrieval pipeline can then by integrated into further PyTerrier pipelines due to the assured compatibility by the TIREx framework.

Listing 3.4 illustrates how any re-rankers submitted to experiments on TIREx can be reused and integrated into any external PyTerrier pipeline. In this case, an initial full-ranking is executed using PyTerrier's `BatchRetrieve` function. Subsequently, the re-ranker is downloaded via the `reranker` method processing the software identifier and is then chained as a subsequent component behind the full-ranker.

Listing 3.5 demonstrates how run files derived from experiments can be downloaded and utilized in PyTerrier pipelines through the `from_submission` method. Since run files serve as input to subsequent steps in standard retrieval pipelines without the need for their originating software, they can be used directly as a component in PyTerrier pipelines, just as they can serve as input for any other software that processes run files. In this case, no software needs to be re-executed, saving time and resources. Any stage of a retrieval pipeline, that is cached by its resulting run file, can thus be accessed and built upon in post-hoc experiments (e.g., with different re-rankers). This method is particularly beneficial for efficient large-scale experiments.

In the next chapter, I discuss several large-scale evaluations that we conducted, making extensive use of the `from_submission` method after caching the runs of 50 retrieval systems across 31 test collections with TIREx.

Chapter 4

Large-Scale Experiments with TIREx

In this chapter, I present three post-hoc experiments conducted on data gathered by the TIREx framework, demonstrating TIREx capabilities to facilitate large-scale evaluations and generalization research. To provide an understanding of the basis for all subsequent studies, I first provide a brief overview of the initial data collection and outline both the diversity and scope of the data pool from which all of our experiments are derived. Subsequently, I present a study that evaluates the replicability of system preferences of 50 retrieval systems across 31 test collections with respect to the preferences observed on the TREC Deep Learning Track 2019 [37]. Further, I report the evaluation of feature-rich learning-to-rank pipelines that we constructed over 6 ClueWeb TREC web collections. In this study, we make use of the PyTerrier implementation of TIREx to collect and implement 43 features from TIRA submissions using only three lines of code. Finally, I conclude with an extensive rank-fusion study we performed on a subset of the accumulated data, in which we comparatively evaluate all possible fusions of runs from 49 different retrieval systems over a set of 25 test collections.

4.1 Initial Data Collection with TIREx

For a sufficiently large data pool, suitable to future large-scale retrieval experiments, we imported 31 test collections from 14 distinct corpora into TIREx, reflecting a wide diversion of tasks. We further deployed 50 retrieval approaches, amounting to a total of 51 retrieval systems, and executed them across all test collections, resulting in 1,550 runs.

Table 4.1 provides an overview of all test collections derived from the corpora we used. Collectively, these corpora encompass 1.7 billion documents.

Table 4.1: Available test collections in TIREx.

Corpus			Included Test Collections	
Name	**Docs.**	**Size**	**Collections**	#
Args.me	0.4 m	8.3 GB	Touché 2020–2021 [14, 15]	2
Antique	0.4 m	90.0 MB	QA Benchmark [54]	1
ClueWeb09	1.0 b	4.0 TB	Web Tracks 2009–2012 [25–28]	4
ClueWeb12	731.7 m	4.5 TB	Web Tracks [32, 33], Touché [14, 15]	4
CORD-19	0.2 m	7.1 GB	TREC-COVID [94, 95]	1
Cranfield	1,400	0.5 MB	Fully Judged Corpus [29, 30]	1
Disks4+5	0.5 m	602.5 GB	TREC-7/8 [91, 92], Robust04 [87, 89]	3
Gov	1.2 m	4.6 GB	Web Tracks 2002–2004 [35, 36]	3
Gov2	25.2 m	87.1 GB	TREC TB 2004–2006 [19, 23, 24]	3
Medline	3.7 m	5.1 GB	Trec Genomics, PM [55, 57, 79, 80]	4
MS MARCO	8.8 m	2.9 GB	Deep Learning 2019–2020 [37, 38]	2
NFCorpus	3,633	30.0 MB	Medical LTR Benchmark [16]	1
Vaswani	11,429	2.1 MB	Scientific Abstracts	1
WaPo	0.6 m	1.6 GB	Core 2018	1
Σ = 14 corpora	1.7 b	8.5 TB		31

Individual test collections span from 1,400 to 7,3 million documents and almost all collections originate from shared tasks of evaluation campaigns.

Table 4.2 lists the 51 retrieval models we submitted to TIREx as Docker images. These models derive from four retrieval frameworks: We implemented 24 models from PyTerrier [72], 17 models from BEIR [84], 9 models from PyGaggle [69], and 1 model from ChatNoir [12]. From PyTerrier, our collection includes 20 lexical retrieval models and 3 DuoT5 cross-encoders via the PyTerrier duoT5 plugin. Additionally, we incorporated one late interaction ColBERT model through the ColBERT plugin for PyTerrier. From BEIR we implemented 17 dense retrieval bi-encoder models, leveraging varied SBERT approaches and from PyGaggle, we included eight variants of monoBERT and monoT5 cross-encoders. In addition to the BM25 model from PyTerrier we also included the BM25F model from the search engine ChatNoir. This ElasticSearch-based engine houses all ClueWeb TrecWeb test collections in TIRA. Therefore, ChatNoir served as the primary BM25 full-rank retrieval approach for all ClueWebs, over the PyTerrier BM25 model, given its capability to utilize all fields within these test collections (e.g., `HTTPS`, `body_content_type`, etc.). All other models were exclusively implemented upon the `default_text` field we introduced in `ir_datasets`. Each model, including BM25F from ChatNoir, operated on its default parameters. While ChatNoir's BM25F model exclusively serves as a full-ranker, all PyTerrier's lexical models and all BEIR's models can operate both as full-rankers and re-rankers. All

Table 4.2: Retrieval systems submitted to TIREx.

Framework	Paradigma	Description	Systems
BEIR [84]	Bi-Encoder	Dense Retrieval	17
ChatNoir [12]	Lexical	Elasticsearch Cluster	1
ColBERT@PT [63]	Late Interaction	Pyterrier Plugin	1
DuoT5@PT [78]	Cross-Encoder	Pairwise Transformer	3
PyGaggle [69]	Cross-Encoder	Pointwise Transformer	9
PyTerrier [72]	Lexical	Traditional Baselines	20
$\Sigma = 6 = 4$ frameworks + 2 forks			51

other models, i.e., PyTerrier's colBERT and duoT5 variants, along with all PyGaggle cross-encoder models, are restricted to re-ranking operations only.

Excluding the ClueWeb test collections, which are already indexed in TIRA using ElasticSearch within ChatNoir, we indexed all other test collections using PyTerrier's indexing function with default parameters. For ClueWeb test collections, we utilized ChatNoir's BM25F model for full-ranking retrieval. For all other test collections, we employed the PyTerrier BM25 model for full-ranking results. To ensure a more valid comparison across all test collections, we then applied all 50 retrieval systems as re-rankers to the full-ranking results of each test collection. Since the focus of this thesis is on the post-hoc experiments performed on this baseline data and does not directly relate to the original results derived from them, I do not address the specifics of the computational environment or the results of the original implementations. For details about the computational settings and original results, I refer to Froebe et al. [45].

4.2 Comparing System Preferences across Test Collections

In this post-hoc evaluation, we utilized the data from the 1,550 runs collected with TIREx to assess the replicability of system preferences across different test collections. By "system preference", we mean the extent to which one retrieval system outperforms another on a specific test collection for a given metric. For instance, within our retrieval results, DLH has an nDCG@10 score of 0.806 on MS MARCO Passage TREC Deep Learning Track 2019, while BM25 holds an nDCG@10 score of 0.795 for the same collection. Therefore DLH is more effective than BM25 for nDCG@10 on Deep Learning track 2019. The degree of this preference can be quantified using various metrics. For our case study, we compiled system preferences between all possible pairs of the

50 retrieval models we implemented, focusing on their performances on TREC Deep Learning track 2019 using the nDCG@10 metric. We then assessed the replicability of these preferences across all other 30 test collections imported into TIREx using `repro_eval` [18].

For any given system preference, we designate the system with the lower nDCG@10 score on TREC Deep Learning track 2019 as the baseline and the one with the higher score as the advanced system. For instance, as previously described, when DLH outperforms BM25 on TREC Deep Learning track 2019, we consider BM25 the baseline and DLH the advanced system. Conversely, when comparing BM25 to DirichletLM, which helds a lower score, BM25 acts as the advanced system to DirichletLM. With `repro_eval`, we evaluate the replicability of these system preferences on a new collection along two dimensions: *Effect Ratio* and *Delta Relative Improvement*. These dimensions are detailed by Breuer et al. [17] with the notation as follows:

Let a and b be runs r, where a-run represents the advanced run and b-run represents the baseline run on an original collection C. Likewise, a' and b' are replicated runs r' performed on a new collection D. M represents any information retrieval evaluation metric. The topics of the original collection are described by $j \in \{1, \ldots, n^C\}$, while the topics of the new collection are described by $j \in \{1, \ldots, n^D\}$. M_j^C denotes the score of a run r on the original collection C with respect to the metric M and topic j. $\overline{M^C(r)}$ is the average score computed across all topics in collection C for run r.

Effect Ratio (ER). As defined in equation (4.2), the *Effect Ratio* quantifies the overall effect of the replication. It is calculated by dividing the mean of the per-topic improvements (4.1) of the replicated experiment by those of the original experiment.

$$\Delta M_j^C = M_j^C(a) - M_j^C(b), \quad \Delta' M_j^D = M_j^D(a') - M_j^D(b') \tag{4.1}$$

$$\mathrm{ER}(\Delta M^D, \Delta M^C) = \frac{\overline{\Delta M^D}}{\overline{\Delta M^C}} = \frac{\frac{1}{n_D}\sum_{j=1}^{n_D} \Delta M_j^D}{\frac{1}{n_C}\sum_{j=1}^{n_C} \Delta M_j^C} \tag{4.2}$$

Delta Relative Improvement (DRI). As defined in Equation (4.4), the *Delta Relative Improvement* measures the difference in relative improvement between the original and replicated experiments. It represents the difference between the relative improvements (4.3) observed in the original run of the original experiment and those in the replicated experiment.

$$\mathrm{RI} = \frac{\overline{M^C(a)} - \overline{M^C(b)}}{\overline{M^C(b)}}, \mathrm{RI}' = \frac{\overline{M^D(a')} - \overline{M^D(b')}}{\overline{M^D(b')}} \tag{4.3}$$

$$\Delta\text{RI}(\text{RI}, \text{RI}') = \text{RI} - \text{RI}' \tag{4.4}$$

For `repro_eval`, a replication is deemed successful, to varying extents, if the overall effect observed between the pair of runs can also be identified on the new collection. The *Effect Ratio* quantifies the extent to which the improvement of the advanced system over the baseline can be mirrored on the new collection. An ideal ER is 1, which represents a perfect replication. Ratios exceeding 1 indicate an even more pronounced improvement during replication, whereas values between 0 and 1 suggest the improvement was less pronounced on the new collection. ER below 0 point to unsuccessful replications as they imply the advanced system performed worse than its baseline on the new collection. The *Delta Relative Improvement* quantifies the difference in improvement between the advanced system and the baseline and typically falls within the [-1, 1] interval. A value of 0 signifies a perfect replication. Values less than 0 suggest a more significant relative improvement of the advanced system on the new collection. Conversely, values greater than 0 but less than 1 indicate a diminished relative improvement. It's possible to witness negative improvements, especially with higher values. This implication, however, isn't directly evident from this measure alone. Therefore, our primary attention centers on the *Effect Ratio*, and we consider any values above 0 as indicative of successful replication. This is because the advanced model's improvement over the baseline is replicated, irrespective of the exact magnitude.

Table 4.3 presents the outcomes of our replicability analysis using `repro_eval`. We display the percentage of successfully replicated system preferences, along with the 25%, 50%, and 75% quantiles for both the *Effect Ratio* and the *Delta Relative Improvement*. The test collections are ordered by the success rate of replicating system preferences observed in the TREC Deep Learning Track 2019 based on ER scores above 0. We highlight the top five test collections with the highest success rate and subsequently continue in increments of five.

Unsurprisingly, replications related to the TREC Deep Learning Track of 2020 are impressive: an 85% success rate, a median ER nearing 0.9, and a DHI close to zero signifying near-perfect replicability (i.e., an ER score of 1 and a DHI of 0). The Touché 2020 Task 2 shows comparable replicability with an 81% success rate, though there's a noticeable drop in median effectiveness, even as the median DHI remains close to zero. For test collections such as Touché 2021 Task 1, Web Track 2004, and CORD-19, replicability is successful in approximately 70% of the instances. However, the median ER sees further decline, with CORD- 19 being a notable exception, boasting a second-best score of 0.4. Despite this, CORD- 19 exhibits inconsistent scores in the first and third quantiles, suggesting a high variability in replicability outcomes

Table 4.3: Replicability results for system preferences from TREC DL 2019 to selected test collections. Includes the success rate in percent (Effect Ratio > 0) and the 25%, 50%, and 75% quantiles for the Effect Ratio and Delta Relative Improvement.

Test Collection	Rank	Succ.	Effect Ratio			Rel. Impr.		
			25 %	50 %	75 %	25 %	50 %	75 %
TREC DL 2020	1	85.2	0.62	0.88	1.06	-0.02	0.02	0.08
Touché 20 (Task 2)	2	81.0	0.17	0.37	0.63	-0.07	0.05	0.15
Touché 21 (Task 2)	3	72.6	-0.06	0.24	0.44	0.01	0.1	0.22
Web Track 2004	4	72.1	-0.03	0.16	0.47	0.01	0.15	0.32
CORD-19	5	70.0	-0.15	0.4	1.75	-0.07	0.12	0.38
Terabyte 2006	10	62.1	-0.26	0.15	1.1	-0.04	0.14	0.44
TREC PM 2017	15	53.4	-0.38	0.05	1.2	-0.3	0.2	0.59
Terabyte 2005	20	42.2	-0.53	-0.15	0.8	-0.01	0.24	0.54
TREC PM 2018	25	33.2	-1.54	-0.51	0.87	-0.15	0.58	1.04
Cranfield	30	28.8	-0.02	-0.01	0.0	0.08	0.29	0.59

for this test collection. At the first quantile, replications across all three test collections tend to falter. Specifically, advanced models from the TREC Deep Learning Track 2019 often underperform compared to their respective baselines. This downward trend becomes more pronounced as we move down the list, impacting even the median values for the latter half of the examined test collections. As a result, for half of these test collections, successful replicability - defined broadly as the consistent performance relation between an advanced model and its baseline, without considering the magnitude of improvement — becomes elusive on average.

4.3 Feature-Full Learning-to-Rank Study

In this post-hoc experiment, we constructed and evaluated a comprehensive learning-to-rank (LTR) pipeline using PyTerrier, building on the extensive resources previously obtained using TIREx. The pipeline incorporates 43 re-rankings as features for training LTR models. Utilizing the PyTerrier implementation of TIREx, we gathered all full-ranking results from the ClueWeb

Listing 4.1: Code snippet for fetching features with the PyTerrier integration from TIREx and their integration into a PyTerrier pipeline.

```
'''
  TASK = ir-benchmarks
  TEAM = tira-ir-starter
  datasets = array of dataset-IDs
  softwares = array of software names
'''

first_stage = tira.pt.from_submission(f'{TASK}/{TEAM}/{chat_noir}',
    datasets=TASKS)
rerankers = [tira.pt.from_submission(f'{TASK}/{TEAM}/{software}') for
    software in softwares]

features = first_stage >> (reduce(lambda x, y: x**y, rerankers))
```

TREC-Web collections from 2009 to 2014. These results were previously obtained using ChatNoir's BM25F implementation and were collected using the `from_submission` method available in the TIRA API-client, which automatically made them PyTerrier-ready (Listing 4.1). We then employed the same method to collect existing re-ranking results from 43 different re-rankers for each of the test collections. From this set of re-ranking data, we assembled a feature-rich pipeline in PyTerrier to serve as a training asset for LTR models. The pipeline was build using PyTerrier's Feature Union operator, as demonstrated in Listing 4.1, and computed 43 individual features from the gathered re-ranking results.

In our experiment, we used six test collections that span from ClueWeb's TREC-Web 2009 to TREC-Web 2014. We created six tasks to evaluate a single learning-to-rank model on each collection. Each task incorporated a test set from its specific collection, a training set compiled from four of the other collections, and a validation set derived from the remaining sixth collection, manually assigned by us. For example, when evaluating an LTR model on the TREC-Web 2009 collection, we assigned TREC-Web 2009 data to serve as the test set, data from TREC-Web 2011 to 2014 to made up the training set, and TREC-Web 2010 data to use for validation. Detailed information about how we distributed the data for these tasks is presented in Table 4.4.

We chose LightGBM [61] as the LTR model for the experiment, taking advantage of PyTerrier's seamless integration with the LightGBM learning library[1]. To enhance model performance, we trained multiple variations of each task-specific model by running a custom grid search on different hyperparameters. The most promising candidates were then selected based on their nDCG@10 scores on the respective validation data. For the grid search key

[1] `https://lightgbm.readthedocs.io/en/latest/Python-Intro.html`

Table 4.4: LTR-tasks with their data distribution for testing, validation and training.

Task	Test	Validate	Train
1	Web Track 2009	Web Track 2010	Web Track 2011, Web Track 2012, Web Track 2013, Web Track 2014
2	Web Track 2010	Web Track 2009	Web Track 2011, Web Track 2012, Web Track 2013, Web Track 2014
3	Web Track 2011	Web Track 2010	Web Track 2009, Web Track 2012, Web Track 2013, Web Track 2014
4	Web Track 2012	Web Track 2011	Web Track 2009, Web Track 2010, Web Track 2013, Web Track 2014
5	Web Track 2013	Web Track 2012	Web Track 2009, Web Track 2010, Web Track 2011, Web Track 2014
6	Web Track 2014	Web Track 2013	Web Track 2009, Web Track 2010, Web Track 2011, Web Track 2012

parameters, we varied the number of iterations (`num_iterations`), the learning rate (`learning_rate`), and the number of leaves (`num_leaves`). Specific values for these parameters are detailed in table 4.5. Any parameters not explicitly mentioned were left at their default settings. This approach resulted in a pool of 36 distinct models for each task.

Table 4.6 compares the performance of the optimal model candidates on their respective test collections against the benchmark scores from the leaderboards[2] of the ir-benchmarks TIRA task. The model evaluated on the 2014 TREC-Web collection showed suboptimal performance, ranking in the bottom percentile of the leaderboard. Models for TREC-Web 2011 and TREC-Web 2013 yielded median performance and ranked in the middle percentile of the leaderboard. In contrast, the models for TREC-Web 2009, TREC-Web 2012, and TREC-Web 2010 performed comparatively better and achieved top ten rankings, with the model for TREC-Web 2010 even ranking fourth. Despite

[2] `https://www.tira.io/task/ir-benchmarks`

Table 4.5: LightGBM parameter setting with Grid Search

Hyperparameter	Values
task	train
num_leaves	[7, 500, 1000]
objective	lambdarank
metric	ndcg
eval_at	[10]
learning_rate	[0.1, 0.2, 0.3, 0.4]
importance_type	gain
num_iterations	[100, 500, 1000]

Table 4.6: nDCG@10 scores of each selected LTR model, compared to the best scores on the ir-benchmark leaderboards (as of 09.09.2023).

Task	Test Set	nDCG@10	ir-benchmarks	
			Best Score	Leaderboard Pos.
1	Web Track 2009	0.147	0.229	9/53
2	Web Track 2010	0.229	0.294	4/54
3	Web Track 2011	0.249	0.289	32/56
4	Web Track 2012	0.160	0.192	6/53
5	Web Track 2013	0.239	0.271	15/53
6	Web Track 2014	0.223	0.340	48/53

these variations, the aggregate data suggest that feature-rich LTR pipelines, as implemented in this study, lack consistent effectiveness across test collections, even when they derive from the same corpora, and do not achieve leading status in any of the evaluations. Several external factors may have limited the study's efficacy, rendering the results inconclusive. These include a fixed split between test and validation data, insufficient tuning of hyperparameters due to a too simple grid search, and indiscriminate feature inclusion. These variables could undermine the model's robustness and generalizability.

4.4 Rank-Fusion Study

In this post-hoc experiment, we applied and investigated rank-fusion for a comparative evaluation against single system retrieval on a large scale. Fusion algorithms integrate different ranked lists into a single ranked list, mitigating the limitations of each ranking while leveraging their strengths in hopes of

Table 4.7: Distribution of retrieval systems by their paradigms used in the rank-fusion study.

Framework	Paradigma	Systems
BEIR	Bi-Encoder	16
ColBERT@PT	Late Interaction	1
DuoT5@PT	Cross-Encoder	3
PyGaggle	Cross-Encoder	9
PyTerrier	Lexical	20
System count:		49

Table 4.8: Summary of test collections used in the rank-fusion study.

Corpus			Included Test Collections	
Name	**Docs.**	**Size**	**Collections**	#
Args.me	0.4 m	8.3 GB	Touché 2020–2021 [14, 15]	2
ClueWeb09	1 b	4.0 TB	Web Tracks 2010–2012 [26–28]	3
ClueWeb12	731.7 m	4.5 TB	Web Tracks [32, 33], Touché [14, 15]	4
CORD-19	0.2 m	7.1 GB	TREC-COVID [94, 95]	1
Disks4+5	0.5 m	602.5 GB	TREC-7/8 [91, 92], Robust04 [87, 89]	3
Gov	1.2 m	4.6 GB	Web Tracks 2002–2004	3
Gov2	25.2 m	87.1 GB	TREC TB 2004–2006	3
Medline	3.7 m	5.1 GB	Trec Genomics, PM	3
MS MARCO	8.8 m	2.9 GB	Deep Learning 2019–2020	2
WaPo	0.6 m	1.6 GB	Core 2018	1
Σ = 10 corpora	1.7 b	8.5 TB		25

improving overall performance.

In this study, we generated fused runs with the ranx library for Python [5], combining every possible diverse pair of a total of 49 different retrieval systems, using the Reciprocal Rank Fusion algorithm (RRF) [34]. This resulted in a total of 1176 fused runs. All runs were assessed on their mean nDCG@10 score based on their runs on 25 test collections. The distribution of retrieval systems according to their underlying paradigms is detailed in Table 4.7. Specifically, we employed 20 lexical models, 16 bi-encoder models, 12 cross-encoder models, and one late interaction model. Details of the test collections used for the evaluation can be found in Table 4.8.

For each pair of systems, we calculated the average nDCG@10 score for both the individual systems and their combined fusion. To statistically validate these results, we conducted a two-tailed paired t-test for each system's score in comparison to the fusion score, using SciPy [86]. The null hypothe-

Table 4.9: Distribution of fusion success by count and percentage.

Fusions	#	**in %**
Total	1176	100.0
Successful	946	80.4
Successful significant	679	57.7
Failed	230	19.6
Failed significant	90	7.7

sis for this test assumes that the average values for the two related samples are identical, therefore a significant difference between the system and fusion scores is indicated by $p \leq 0.05$. These values were adjusted using a Bonferroni correction factor of 1176. We also computed Cohen's D to quantify the effect size between each system's score and the fused score. We consider a fusion successful, if its score outperforms the scores of both individual systems. In such cases, the system with the higher score is referred to as the "superior system," while the system with the lower score is referred to as the "helper system".

In Table 4.9, we present the distribution of successful fusions versus failed fusions, further filtered by statistical significance. Out of the initial 1176 fusions, 80% were successful. When filtered for statistical significance via t-test, 679 out of 946 successful fusions remained, signifying that 70% of all successful fusions resulted in a significant improvement in effectiveness. Conversely, out of the total 230 failed fusions, only 90 remained statistically significant, accounting for 39% of all failed fusions. This suggests that losses in effectiveness are statistically less likely to be significant when systems are fused.

Each of the 49 retrieval systems participates in a total of 48 different fusions with other systems. We define an effective fusion partner as the helper system in a successful fusion, as it elevates an already superior system to become even more effective. Table 4.10 provides a detailed list of the most common helper systems, in terms of their nDCG@10 values. The table lists the total number of cases in which each system participates as a helper. It also shows the percentage representation of each system's role as a helper relative to all fusion combinations in which it participated. In addition, the underlying paradigm of each helper system is indicated. The list is ranked based on the frequency with which each system appears as a helper. We truncated the list to exclude systems with a frequency below 50%, as these tend to have subpar performance on average. Although bi-encoding algorithms dominate the ranking, three out of four paradigms are represented in the top 5. $\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$

Table 4.10: Ranking of the most frequent helper systems. Includes the total count of the system as a helper system in all of its 48 fusions, the percentage as a helper system in all of its 48 fusions and the paradigm underlying the helper system.

Helper System	# in Comb.	% in Comb.	Paradigma
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	38 / 48	79.2	bi-encoder
DuoT5 Top-25	38 / 48	79.2	cross-encoder
DirichletLM	36 / 48	75.0	lexical
$\text{SBERT}_{\text{multi-qa-MiniLM-L6-dot-v1}}$	36 / 48	75.0	bi-encoder
ANCE Base Cosine	36 / 48	75.0	bi-encoder
ANCE Base Dot	34 / 48	70.8	bi-encoder
$\text{SBERT}_{\text{msmarco-MiniLM-L6-cos-v5}}$	34 / 48	70.8	bi-encoder
$\text{SBERT}_{\text{multi-qa-MiniLM-L6-cos-v1}}$	33 / 48	68.8	bi-encoder
$\text{SBERT}_{\text{msmarco-MiniLM-L12-cos-v5}}$	33 / 48	68.8	bi-encoder
$\text{MonoBERT}_{\text{Small-MS-MARCO-10k}}$	32 / 48	66.7	cross-encoder
$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	32 / 48	66.7	bi-encoder
$\text{TASB}_{\text{msmarco-distilbert-base-cos}}$	31 / 48	64.6	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-cos}}$	30 / 48	62.5	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-cos-v5}}$	30 / 48	62.5	bi-encoder
DFIC	27 / 48	56.2	lexical
DFIZ	26 / 48	54.2	lexical

stands out as the most effective bi-encoder, DuoT5 Top-25 as the most effective cross-encoder and DirichletLM represents the most effective lexical algorithm.

In Figure 4.1, we extend our evaluation to include the effect sizes by which the fusion score surpasses the individual system scores. Each helper system is plotted on a coordinate plane: the x-axis reflects the effect size by which the fusion surpasses the helper system's performance, and the y-axis reflects the median effect size by which the fusion improves upon the superior systems in the pairings. Each dot on the plot represents an aggregated set of system pairings featuring a distinct helper system. These dots are color-coded based on the paradigm of the helper system and are sized proportionally to the number of aggregated system pairs.

The insights from Figure 4.1 serve to visualize the findings presented in Table 4.10. Specifically, bi-encoders appear to be more effective as helper systems compared to lexical models. On average, they contribute more substantially to enhancing the effect size by which fusions improve on their superior system partners and are also more frequently observed in the role of a helper system. In contrast, cross-encoders do not often participate as helper systems and when

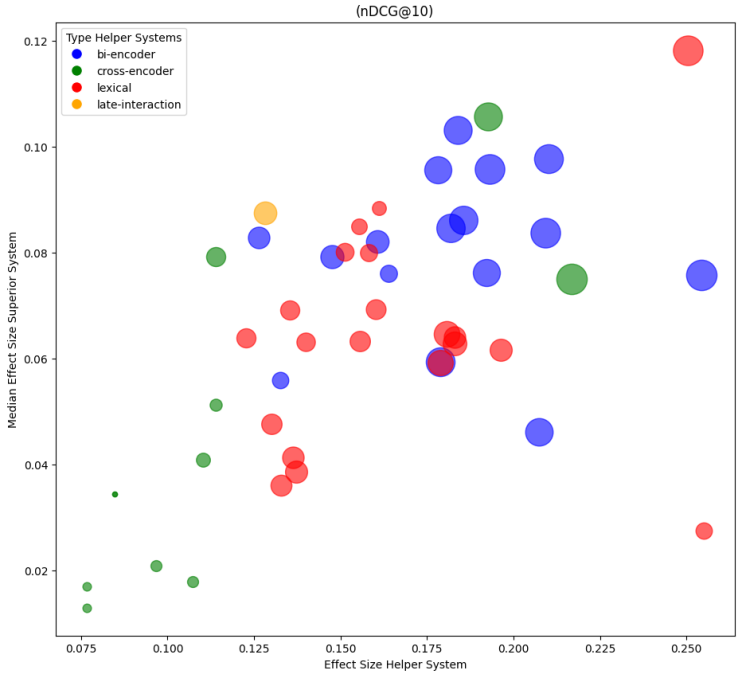

Figure 4.1: Effectiveness of helper systems when fused with superior systems, color-coded by their underlying paradigms. Each dot represents a unique retrieval system, with its size proportional to the frequency of its appearance as a helper system. The y-axis shows the median effect size by which the fusion improved over the superior systems, while the x-axis represents the effect size by which the fusion improved over the helper system.

they do their impact is generally modest, with the exception of three cases. Despite the generally weaker performance of lexical models, one specific lexical system stands out as the most effective helper system across all paradigms.

In Tables A.1, A.2 and 4.11, the focus lies on the most effective helper systems. The first two tables reports the optimal partner for each superior system across all fusions (seperated into two tables for formatting issues). The second table aggregates this data, listing the systems that most frequently emerge as the optimal partner systems. It ranks them according to their frequency and also includes the median effect size by which the fusion improves on their superior system counterparts. In this analysis of the best helper systems, only 11 unique systems are identified. Among these, bi-encoders are still strongly represented. However, DirichletLM in particular stands out with overwhelming frequency as the overall most effective partner. Moreover, it consistently promotes a high median effect size by which its fusions outperform

Table 4.11: Ranking of top-performing helper systems. includes their frequency of being identified as the optimal helper, average Effect Size enhancements by which fusions surpass the superior partner systems, and paradigm category. (nDCG@10)

Best Helper System	#	Median Effect Size	Paradigma
DirichletLM	22	0.171	lexical
$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	6	0.140	bi-encoder
ANCE Base Dot	4	0.135	bi-encoder
IFB2	3	0.090	lexical
$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	3	0.144	bi-encoder
$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	3	0.145	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	2	0.214	bi-encoder
DuoT5 Top-25	2	0.182	cross-encoder
ANCE Base Cosine	1	0.131	bi-encoder
Hiemstra_LM	1	0.003	lexical
$\text{MonoBERT}_{\text{Small-MS-MARCO-10k}}$	1	0.183	cross-encoder

the participating superior systems.

Finally, we further evaluated the best helper systems at the paradigm level, analyzing which helper systems support superior systems of a given paradigm in successful fusions. In Figures 4.2 through 4.5, we evaluate helper systems of all paradigms paired with superior systems of just one uniform paradigm. Figure 4.2 focuses on helper systems paired with lexical models as the superior systems. Figure 4.3 deals with bi-encoding models, Figure 4.4 with cross-encoding models, and Figure 4.5 with the single late interaction model. These plots present an overview of which paradigms are most effective at improving the performance of superior systems that are uniformly from a given paradigm within fusions.

To complement the visual data in the figures, Tables 4.12 through 4.15 present the aggregated data on the optimal helper systems within each paradigm, sorted by how frequently they appear. These tables provide additional insights about the specific systems that have the most impact as helper systems when fused with superior systems of the respective paradigms. The aggregations suggests that lexical models benefit most when fused with bi-encoding algorithms. Conversely, bi-encoders, cross-encoders, and the single late interaction model generally show the most improvement when fused with lexical approaches. Although fusions within the same paradigm often yield successful results, their impact is typically marginal. This suggests that superior performance gains are more likely to be achieved by fusing systems from

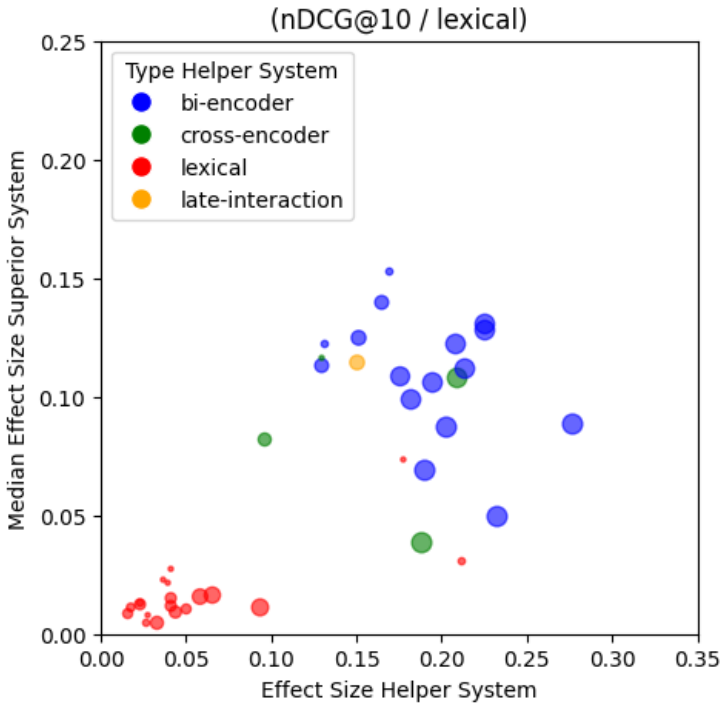

Figure 4.2: Effectiveness of helper systems when fused with superior lexical systems. Details as in in Figure 4.1

different paradigms, possibly because systems within the same paradigm tend to have overlapping strengths and weaknesses. Interestingly, DirichletLM, a system generally considered to be rather low performing, emerges as the predominant choice for improving the performance of more sophisticated systems across almost all paradigms. Statistically it far outperforms all other options and often seems to be the most effective system to fuse with. Given its computational efficiency, DirichletLM appears to be a valuable asset in many cases for improving the performance of high-end, complex systems through rank fusion.

Table 4.12: Ranking of top-performing helper systems for lexical systems. Details as in Table 4.11

Best Helper System	#	Median Effect Size	Paradigma
$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	6	0.140	bi-encoder
$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	3	0.145	bi-encoder
ANCE Base Dot	3	0.140	bi-encoder
$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	2	0.153	bi-encoder
ANCE Base Cosine	1	0.131	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	1	0.214	bi-encoder
IFB2	1	0.022	lexical
Hiemstra_LM	1	0.003	lexical
$\text{MonoBERT}_{\text{Small-MS-MARCO-10k}}$	1	0.183	cross-encoder

Table 4.13: Ranking of top-performing helper systems for bi-encoders. Details as in Table 4.11

Best Helper System	#	Median Effect Size	Paradigma
DirichletLM	12	0.206	lexical
DuoT5 Top-25	2	0.182	cross-encoder
IFB2	2	0.109	lexical

Table 4.14: Ranking of top-performing helper systems for cross-encoders. Details as in Table 4.11

Best Helper System	#	Median Effect Size	Paradigma
DirichletLM	9	0.120	lexical
$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	1	0.111	bi-encoder
ANCE Base Dot	1	0.126	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	1	0.214	bi-encoder

Table 4.15: Ranking of top-performing helper systems for late-interaction systems. Details as in Table 4.11

Best Helper System	#	Median Effect Size	Paradigma
DirichletLM	1	0.176	lexical

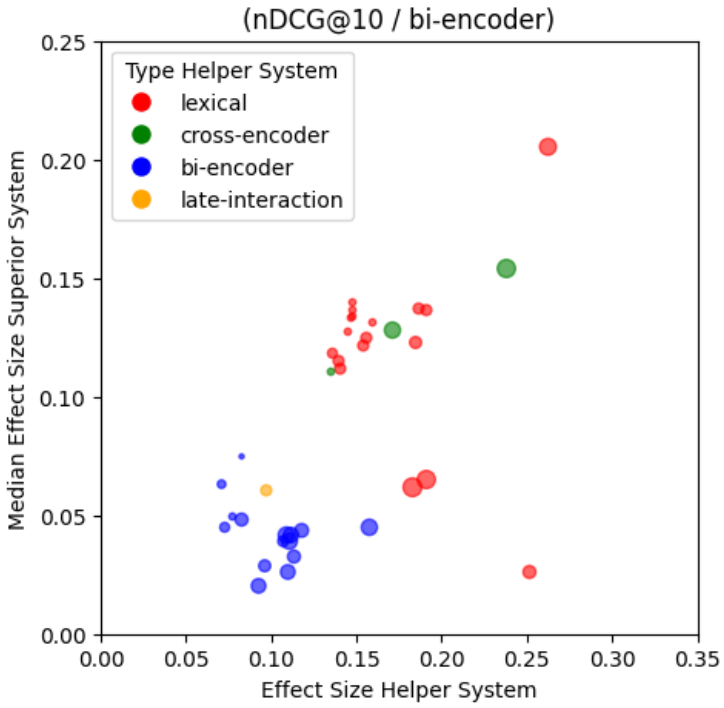

Figure 4.3: Effectiveness of helper systems when fused with superior bi-encoder systems. Details as in in Figure 4.1

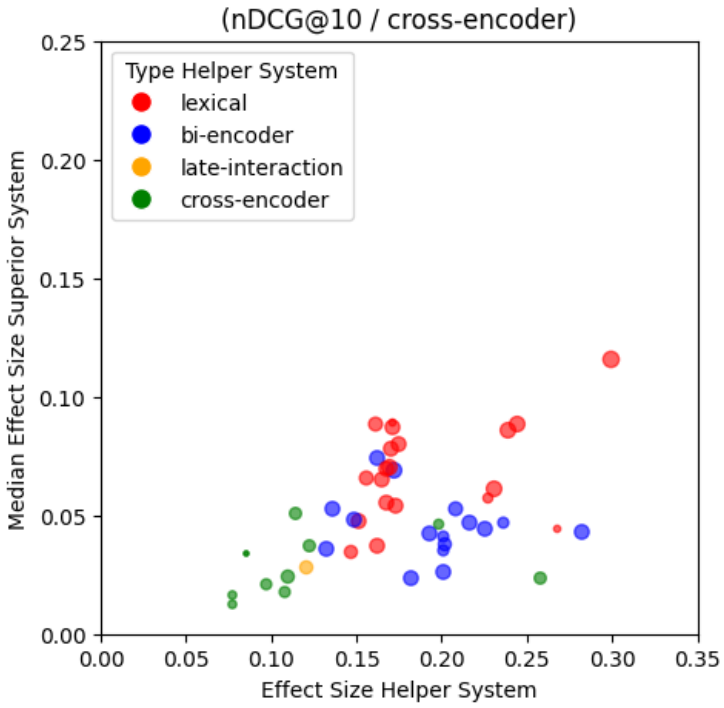

Figure 4.4: Effectiveness of helper systems when fused with superior cross-encoder systems. Details as in in Figure 4.1

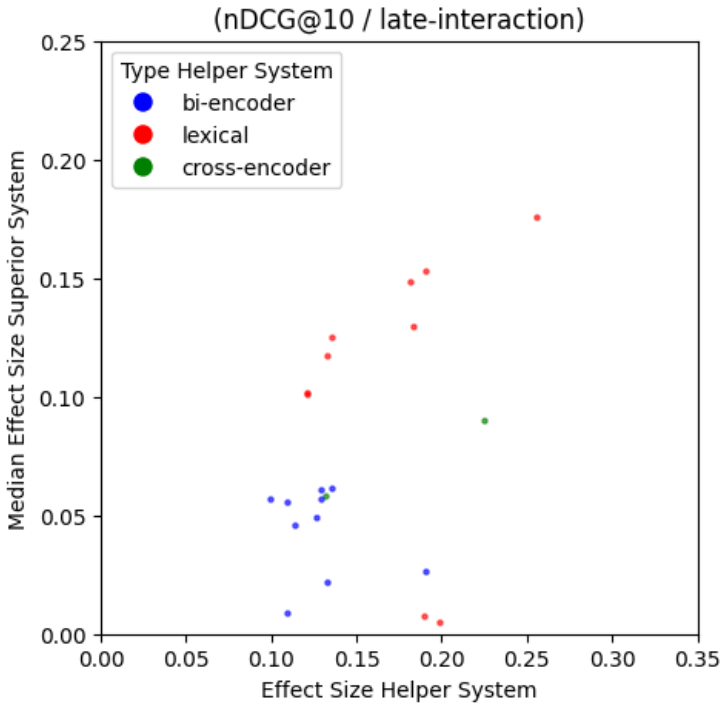

Figure 4.5: Effectiveness of helper systems when fused with superior late-interaction systems. Details as in in Figure 4.1

Chapter 5

Conclusion

In Chapter 3 I present how we integrate `ir_datasets`, PyTerrier, and `ir_measures` with TIRA to form The Information Retrieval Experiment Platform, a framework designed to facilitate replicability and encourage research aimed at generalization. By differentiating tasks and standardizing the input and output for each component, while ensuring compatibility with PyTerrier interfaces, we make both software and test collections interchangeable. TIRA's archiving abilities, supported by Docker and self-contained Git repositories, enable further post-hoc evaluations even beyond these frameworks. This makes experiments not only reusable but also scalable. In Chapter 4, we make use of TIREx's capabilities to conduct three large-scale post-hoc experiments. These experiments utilize data gathered from 50 retrieval systems across 31 test collections that were initially integrated into TIREx. Our first evaluation provides insights into the replicability of system preferences. We observe the preferences of all pairs of 50 retrieval systems on the Deep Learning Web Track 2019 and replicate them across 30 different test collections. We then compared the results by measuring the Effect Ratio and the change in Relative Improvement compared to the original combinations. These analyses report insights into how system preferences differ on other test collections. We further deployed 6 learning-to-rank pipelines constructed from the ClueWeb test collections of 2009-2014. These pipelines were enhanced with features derived from 43 of the 50 retrieval systems that were previously evaluated on TIREx and could be seamlessly integrated into PyTerrier pipelines using TIREx's PyTerrier implementation. The resulting scores are compared with those on TIRA's complementary leaderboards, which are populated with results from the initial evaluations of all retrieval systems. Finally, we evaluate all possible fusions among 49 retrieval systems using Reciprocal Rank Fusion, comparing their mean nDCG@10 scores from runs on 25 test collections. This analysis offers statistical insights into the potential effectiveness of fusion techniques,

specifically highlighting trends regarding how the underlying paradigms of the systems influence the fusion's performance. The clear preference for DirichletLM as the statistically most effective fusion partner is the notable finding with which this final study concludes.

Appendix A

Appendix

Table A.1: List of all evaluated systems that appeared as a superior system and the optimal partner systems that improved their score the most within a fusion (nDCG@10).
Note: Table continues in Table A.2 due to formatting constraints.

System (superior)	Best Partner System	Effect Size	Paradigma
MonoT5 3b	DirichletLM	0.056	lexical
MonoT5 Large	DirichletLM	0.096	lexical
$\text{MonoBERT}_{\text{Large-MS-MARCO-10k}}$	DirichletLM	0.096	lexical
MonoBERT Base	DirichletLM	0.133	lexical
MonoT5 Base	DirichletLM	0.137	lexical
$\text{MonoBERT}_{\text{Large-Finetune-Only}}$	DirichletLM	0.116	lexical
MonoBERT Large	DirichletLM	0.120	lexical
$\text{DuoT5}_{\text{3b-ms-marco Top-25}}$	$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	0.111	bi-encoder
InB2	$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	0.144	bi-encoder
XSqrA_M	$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	0.162	bi-encoder
MonoBERT Small	DirichletLM	0.151	lexical
$\text{SBERT}_{\text{msmarco-bert-base-dot-v5}}$	DirichletLM	0.149	lexical
$\text{SBERT}_{\text{multi-qa-mpnet-base-cos-v1}}$	DirichletLM	0.166	lexical
In_expB2	$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	0.140	bi-encoder
In_expC2	$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	0.135	bi-encoder
InL2	$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	0.140	bi-encoder
TF_IDF	$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	0.143	bi-encoder
DPH	$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	0.141	bi-encoder
$\text{DuoT5}_{\text{base-10k-ms-marco Top-25}}$	ANCE Base Dot	0.126	bi-encoder
LGD	$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	0.158	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-dot-v5}}$	DirichletLM	0.165	lexical
$\text{SBERT}_{\text{multi-qa-distilbert-cos-v}}1$	DirichletLM	0.178	lexical
Js_KLs	ANCE Base Dot	0.131	bi-encoder
$\text{TASB}_{\text{msmarco-distilbert-base-dot}}$	DirichletLM	0.201	lexical
ColBERT	DirichletLM	0.176	lexical
BM25	$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	0.145	bi-encoder
DFR_BM25	$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	0.144	bi-encoder
DFReeKLIM	ANCE Base Cosine	0.131	bi-encoder
DFRee	ANCE Base Dot	0.140	bi-encoder

Table A.2: Table A.1 (Continued)

System (superior)	Best Partner System	Effect Size	Paradigma
DFIZ	ANCE Base Dot	0.193	bi-encoder
DFIC	$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	0.192	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-cos}}$	DirichletLM	0.197	lexical
PL2	$\text{MonoBERT}_{\text{Small-MS-MARCO-10k}}$	0.183	cross-encoder
$\text{SBERT}_{\text{multi-qa-distilbert-dot-v1}}$	DirichletLM	0.241	lexical
$\text{SBERT}_{\text{msmarco-MiniLM-L12-cos-v5}}$	DirichletLM	0.210	lexical
ANCE Base Dot	DirichletLM	0.252	lexical
ANCE Base Cosine	DirichletLM	0.246	lexical
$\text{SBERT}_{\text{multi-qa-MiniLM-L6-cos-v1}}$	DirichletLM	0.242	lexical
$\text{MonoBERT}_{\text{Small-MS-MARCO-10k}}$	DirichletLM	0.264	lexical
$\text{SBERT}_{\text{multi-qa-MiniLM-L6-dot-v1}}$	DirichletLM	0.251	lexical
DirichletLM	$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	0.214	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-cos-v5}}$	DuoT5 Top-25	0.183	cross-encoder
$\text{SBERT}_{\text{msmarco-MiniLM-L6-cos-v5}}$	DuoT5 Top-25	0.182	cross-encoder
DuoT5 Top-25	$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	0.214	bi-encoder
$\text{SBERT}_{\text{msmarco-distilbert-base-v3-dot}}$	IFB2	0.128	lexical
$\text{TASB}_{\text{msmarco-distilbert-base-cos}}$	IFB2	0.090	lexical
DLH	IFB2	0.022	lexical
IFB2	Hiemstra_LM	0.003	lexical

Bibliography

[1] Jaime Arguello, Matt Crane, Fernando Diaz, Jimmy Lin, and Andrew Trotman. Report on the SIGIR 2015 workshop on reproducibility, inexplicability, and generalizability of results (RIGOR). *SIGIR Forum*, 49(2): 107–116, 2015. doi: 10.1145/2888422.2888439. URL `https://doi.org/10.1145/2888422.2888439`.

[2] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 601–610. ACM, 2009.

[3] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Evaluatir: an online tool for evaluating and comparing IR systems. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, page 833. ACM, 2009. doi: 10.1145/1571941.1572153. URL `https://doi.org/10.1145/1571941.1572153`.

[4] Leif Azzopardi, Paul Thomas, and Alistair Moffat. cwl_eval: An evaluation tool for information retrieval. In *SIGIR*, 2019.

[5] Elias Bassani and Luca Romelli. ranx.fuse: A python library for metasearch. In Mohammad Al Hasan and Li Xiong, editors, *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 4808–4812. ACM, 2022. doi: 10.1145/3511808.3557207. URL `https://doi.org/10.1145/3511808.3557207`.

[6] Nicholas J. Belkin, A. Cabezas, Colleen Cool, K. Kim, Kwong Bor Ng, Soyeon Park, R. Pressman, Soo Young Rieh, Pamela A. Savage-Knepshield, and Hong (Iris) Xie. Rutgers interactive track at TREC-5.

In Ellen M. Voorhees and Donna K. Harman, editors, *Proceedings of The Fifth Text REtrieval Conference, TREC 1996, Gaithersburg, Maryland, USA, November 20-22, 1996*, volume 500-238 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 1996. URL `http://trec.nist.gov/pubs/trec5/papers/ruint_paper.ps.gz`.

[7] Nicholas J. Belkin, Jose Perez Carballo, Colleen Cool, Diane Kelly, Shinjeng Lin, Soyeon Park, Soo Young Rieh, Pamela A. Savage-Knepshield, and C. Sikora. Rutgers' TREC-7 interactive track experience. In Ellen M. Voorhees and Donna K. Harman, editors, *Proceedings of The Seventh Text REtrieval Conference, TREC 1998, Gaithersburg, Maryland, USA, November 9-11, 1998*, volume 500-242 of *NIST Special Publication*, pages 221–229. National Institute of Standards and Technology (NIST), 1998.

[8] Nicholas J. Belkin, Amymarie Keller, Diane Kelly, Jose Perez Carballo, C. Sikora, and Ying Sun. Support for question-answering in interactive information retrieval: Rutgers' TREC-9 interactive track experience. In Ellen M. Voorhees and Donna K. Harman, editors, *Proceedings of The Ninth Text REtrieval Conference, TREC 2000, Gaithersburg, Maryland, USA, November 13-16, 2000*, volume 500-249 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2000. URL `http://trec.nist.gov/pubs/trec9/papers/rutgers-int.pdf`.

[9] Nicholas J. Belkin, Colleen Cool, J. Jeng, Amymarie Keller, Diane Kelly, Ja-Young Kim, Hyuk-Jin Lee, Muh-Chyun (Morris) Tang, and Xiaojun Yuan. Rutgers' TREC 2001 interactive track experience. In Ellen M. Voorhees and Donna K. Harman, editors, *Proceedings of The Tenth Text REtrieval Conference, TREC 2001, Gaithersburg, Maryland, USA, November 13-16, 2001*, volume 500-250 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2001. URL `http://trec.nist.gov/pubs/trec10/papers/rutgers-interactive-paper.pdf`.

[10] Nicholas J. Belkin, Diane Kelly, G. Kim, Ja-Young Kim, Hyuk-Jin Lee, Gheorghe Muresan, Muh-Chyun (Morris) Tang, Xiaojun Yuan, and Colleen Cool. Rutgers interactive track at TREC 2002. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of The Eleventh Text REtrieval Conference, TREC 2002, Gaithersburg, Maryland, USA, November 19-22, 2002*, volume 500-251 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2002. URL `http://trec.nist.gov/pubs/trec11/papers/rutgers.belkin.pdf`.

[11] Nicholas J. Belkin, Diane Kelly, Hyuk-Jin Lee, Yuelin Li, Gheorghe Muresan, Muh-Chyun (Morris) Tang, Xiaojun Yuan, and Xiao-Min Zhang. Rutgers' HARD and web interactive track experiments at TREC 2003. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*, volume 500-255 of *NIST Special Publication*, pages 532–543. National Institute of Standards and Technology (NIST), 2003. URL `http://trec.nist.gov/pubs/trec12/papers/rutgersu.hard.web.pdf`.

[12] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Elastic chatnoir: Search engine for the clueweb and the common crawl. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*, volume 10772 of *Lecture Notes in Computer Science*, pages 820–824. Springer, 2018. doi: 10.1007/978-3-319-76941-7_83. URL `https://doi.org/10.1007/978-3-319-76941-7_83`.

[13] David C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Commun. ACM*, 28(3):289–299, 1985. doi: 10.1145/3166.3197. URL `https://doi.org/10.1145/3166.3197`.

[14] Alexander Bondarenko, Lukas Gienapp, Maik Fröbe, Meriem Beloucif, Yamen Ajjour, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. Overview of touché 2021: Argument retrieval. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2258–2284. CEUR-WS.org, 2021. URL `https://ceur-ws.org/Vol-2936/paper-205.pdf`.

[15] Alexander Bondarenko, Maik Fröbe, Johannes Kiesel, Shahbaz Syed, Timon Gurcke, Meriem Beloucif, Alexander Panchenko, Chris Biemann, Benno Stein, Henning Wachsmuth, Martin Potthast, and Matthias Hagen. Overview of touché 2022: Argument retrieval. In Guglielmo Faggioli, Nicola Ferro, Allan Hanbury, and Martin Potthast, editors, *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - to - 8th, 2022*, volume 3180 of *CEUR Workshop Proceedings*, pages 2867–2903. CEUR-WS.org, 2022. URL `https://ceur-ws.org/Vol-3180/paper-247.pdf`.

[16] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. A full-text learning to rank dataset for medical information retrieval. In *Proceedings of the European Conference on Information Retrieval (ECIR)*. Springer, 2016.

[17] Timo Breuer, Nicola Ferro, Norbert Fuhr, Maria Maistro, Tetsuya Sakai, Philipp Schaer, and Ian Soboroff. How to measure the reproducibility of system-oriented ir experiments. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 349–358, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3397271.3401036.

[18] Timo Breuer, Nicola Ferro, Maria Maistro, and Philipp Schaer. repro_eval: A python interface to reproducibility measures of system-oriented IR experiments. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, Lecture Notes in Computer Science, pages 481–486. Springer, 2021. doi: 10.1007/978-3-030-72240-1_51.

[19] Stefan Büttcher, Charles L. A. Clarke, and Ian Soboroff. The trec 2006 terabyte track. In *TREC*, 2006.

[20] Ben Carterette. The best published result is random: Sequential testing and its effect on reported effectiveness. In Ricardo Baeza-Yates, Mounia Lalmas, Alistair Moffat, and Berthier A. Ribeiro-Neto, editors, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 747–750. ACM, 2015. doi: 10.1145/2766462.2767812. URL `https://doi.org/10.1145/2766462.2767812`.

[21] Marc-Allen Cartright, Samuel J. Huston, and Henry Feild. Galago: A modular distributed processing and retrieval system. In Andrew Trotman, Charles L. A. Clarke, Iadh Ounis, J. Shane Culpepper, Marc-Allen Cartright, and Shlomo Geva, editors, *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval, OSIR@SIGIR 2012, Portland, Oregon, USA, 16th August 2012*, pages 25–31. University of Otago, Dunedin, New Zealand, 2012.

[22] Kenneth Ward Church and Valia Kordoni. Emerging trends: Sota-chasing. *Natural Language Engineering*, 28(2):249–269, 2022. doi: 10.1017/S1351324922000043.

[23] Charles L. A. Clark, Falk Scholer, and Ian Soboroff. The trec 2005 terabyte track. In *TREC*, 2005.

[24] Charles Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2004 terabyte track. In *TREC*, 2004.

[25] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2009 web track. In *TREC*, 2009.

[26] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Gordon V. Cormack. Overview of the trec 2010 web track. In *TREC*, 2010.

[27] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. Overview of the trec 2011 web track. In *TREC*, 2011.

[28] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. Overview of the trec 2012 web track. In *TREC*, 2012.

[29] Cyril Cleverdon. The cranfield tests on english language devices. *Aslib Proceedings*, 19(6):173–194, 1967. ISSN 0001-253X. doi: 10.1108/eb050097. URL `https://doi.org/10.1108/eb050097`.

[30] Cyril W. Cleverdon. The significance of the cranfield tests on index languages. In Abraham Bookstein, Yves Chiaramella, Gerard Salton, and Vijay V. Raghavan, editors, *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Chicago, Illinois, USA, October 13-16, 1991 (Special Issue of the SIGIR Forum)*, pages 3–12. ACM, 1991. doi: 10.1145/122860.122861. URL `https://doi.org/10.1145/122860.122861`.

[31] Christian S. Collberg. Measuring reproducibility in computer systems research. 2014.

[32] Kevyn Collins-Thompson, Paul Bennett, Fernando Diaz, Charles L. A. Clarke, and Ellen M. Voorhees. Trec 2013 web track overview. In *TREC*, 2013.

[33] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M. Voorhees. Trec 2014 web track overview. In *TREC*, 2014.

[34] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*, pages 758–759. ACM, 2009.

[35] Nick Craswell and David Hawking. Overview of the trec-2004 web track. In *TREC*, 2004.

[36] Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang Wu. Overview of the trec 2003 web track. In *TREC*, 2003.

[37] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen Voorhees. Overview of the trec 2019 deep learning track. In *TREC 2019*, 2019.

[38] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2020 deep learning track. In *TREC*, 2020.

[39] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820, 2020. URL `https://arxiv.org/abs/2003.07820`.

[40] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the TREC 2020 deep learning track. *CoRR*, abs/2102.07662, 2021. URL `https://arxiv.org/abs/2102.07662`.

[41] Emanuele Di Buccio, Nicola Ferro, Donna Harman, Maria Maistro, Gianmaria Silvello, and Giorgio Maria Di Nunzio. Unfolding off-the-shelf ir systems for reproducibility. 08 2015. doi: 10.13140/RG.2.1.3475.0161.

[42] Nicola Ferro and Diane Kelly. SIGIR initiative to implement ACM artifact review and badging. *SIGIR Forum*, 52(1):4–10, 2018. doi: 10.1145/3274784.3274786. URL `https://doi.org/10.1145/3274784.3274786`.

[43] Nicola Ferro, Norbert Fuhr, Kalervo Järvelin, Noriko Kando, Matthias Lippold, and Justin Zobel. Increasing reproducibility in IR: findings from the dagstuhl seminar on "reproducibility of data-oriented experiments in e-science". *SIGIR Forum*, 50(1):68–82, 2016. doi: 10.1145/2964797.2964808. URL `https://doi.org/10.1145/2964797.2964808`.

[44] Nicola Ferro, Norbert Fuhr, Gregory Grefenstette, Joseph A. Konstan, Pablo Castells, Elizabeth M. Daly, Thierry Declerck, Michael D. Ekstrand, Werner Geyer, Julio Gonzalo, Tsvi Kuflik, Krister Lindén, Bernardo Magnini, Jian-Yun Nie, Raffaele Perego, Bracha Shapira, Ian Soboroff, Nava Tintarev, Karin Verspoor, Martijn C. Willemsen, and Justin Zobel. The dagstuhl perspectives workshop on performance modeling and prediction. *SIGIR Forum*, 52(1):91–101, 2018. doi: 10.1145/3274784.3274789. URL `https://doi.org/10.1145/3274784.3274789`.

[45] Maik Fröbe, Jan Heinrich Reimer, Sean MacAvaney, Niklas Deckers, Simon Reich, Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. The information retrieval experiment platform. In Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete, editors, *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2826–2836. ACM, 2023. doi: 10.1145/3539618.3591888. URL https://doi.org/10.1145/3539618.3591888.

[46] Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. Continuous integration for reproducible shared tasks with tira.io. In Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo, editors, *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part III*, volume 13982 of *Lecture Notes in Computer Science*, pages 236–241. Springer, 2023. doi: 10.1007/978-3-031-28241-6_20. URL https://doi.org/10.1007/978-3-031-28241-6_20.

[47] Norbert Fuhr. Salton award lecture information retrieval as engineering science. *SIGIR Forum*, 46(2):19–28, 2012. doi: 10.1145/2422256.2422259. URL https://doi.org/10.1145/2422256.2422259.

[48] Norbert Fuhr. Some common mistakes in IR evaluation, and how they can be avoided. *SIGIR Forum*, 51(3):32–41, 2017. doi: 10.1145/3190580.3190586. URL https://doi.org/10.1145/3190580.3190586.

[49] Norbert Fuhr. Proof by experimentation?: towards better IR research. *SIGIR Forum*, 54(2):2:1–2:4, 2020. doi: 10.1145/3483382.3483385. URL https://doi.org/10.1145/3483382.3483385.

[50] Norbert Fuhr. Proof by experimentation? towards better IR research. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, page 2. ACM, 2020. doi: 10.1145/3397271.3402426. URL https://doi.org/10.1145/3397271.3402426.

[51] Tim Gollub, Benno Stein, Steven Burrows, and Dennis Hoppe. TIRA: configuring, executing, and disseminating information retrieval experiments.

In Abdelkader Hameurlain, A Min Tjoa, and Roland R. Wagner, editors, *23rd International Workshop on Database and Expert Systems Applications, DEXA 2012, Vienna, Austria, September 3-7, 2012*, pages 151–155. IEEE Computer Society, 2012. doi: 10.1109/DEXA.2012.55. URL `https://doi.org/10.1109/DEXA.2012.55`.

[52] Allan Hanbury, Henning Müller, Georg Langs, Marc-André Weber, Bjoern H. Menze, and Tomas Salas Fernandez. Bringing the algorithms to the data: Cloud-based benchmarking for medical image analysis. In Tiziana Catarci, Pamela Forner, Djoerd Hiemstra, Anselmo Peñas, and Giuseppe Santucci, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Visual Analytics - Third International Conference of the CLEF Initiative, CLEF 2012, Rome, Italy, September 17-20, 2012. Proceedings*, volume 7488 of *Lecture Notes in Computer Science*, pages 24–29. Springer, 2012. doi: 10.1007/978-3-642-33247-0_3. URL `https://doi.org/10.1007/978-3-642-33247-0_3`.

[53] Allan Hanbury, Henning Müller, Krisztian Balog, Torben Brodt, Gordon V. Cormack, Ivan Eggel, Tim Gollub, Frank Hopfgartner, Jayashree Kalpathy-Cramer, Noriko Kando, Anastasia Krithara, Jimmy Lin, Simon Mercer, and Martin Potthast. Evaluation-as-a-service: Overview and outlook. *CoRR*, abs/1512.07454, 2015. URL `http://arxiv.org/abs/1512.07454`.

[54] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. ANTIQUE: A non-factoid question answering benchmark. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, volume 12036 of *Lecture Notes in Computer Science*, pages 166–173. Springer, 2020. doi: 10.1007/978-3-030-45442-5_21. URL `https://doi.org/10.1007/978-3-030-45442-5_21`.

[55] William Hersh, Aaron Cohen, Jianji Yang, Ravi Teja Bhupatiraju, Phoebe Roberts, and Marti Hearst. Trec 2005 genomics track overview. In *TREC*, 2007.

[56] William R. Hersh and Paul Over. TREC-8 interactive track. *SIGIR Forum*, 33(2):8–11, 1999. doi: 10.1145/344250.344251. URL `https://doi.org/10.1145/344250.344251`.

[57] William R. Hersh, Ravi Teja Bhuptiraju, Laura Ross, Phoebe Johnson, Aaron M. Cohen, and Dale F. Kraemer. Trec 2004 genomics track overview. In *TREC*, 2004.

[58] Frank Hopfgartner, Allan Hanbury, Henning Müller, Ivan Eggel, Krisztian Balog, Torben Brodt, Gordon V. Cormack, Jimmy Lin, Jayashree Kalpathy-Cramer, Noriko Kando, Makoto P. Kato, Anastasia Krithara, Tim Gollub, Martin Potthast, Evelyne Viegas, and Simon Mercer. Evaluation-as-a-service for the computational sciences: Overview and outlook. *ACM J. Data Inf. Qual.*, 10(4):15:1–15:32, 2018. doi: 10.1145/3239570. URL https://doi.org/10.1145/3239570.

[59] John P. A. Ioannidis. Why most published research findings are false. *PLOS Medicine*, 2(8), 08 2005. doi: 10.1371/journal.pmed.0020124. URL https://doi.org/10.1371/journal.pmed.0020124.

[60] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20 (4):422–446, 2002. doi: 10.1145/582415.582418. URL https://doi.org/10.1145/582415.582418.

[61] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[62] Sadegh Kharazmi, Falk Scholer, David Vallet, and Mark Sanderson. Examining additivity and weak baselines. *ACM Trans. Inf. Syst.*, 34(4): 23:1–23:18, 2016. doi: 10.1145/2882782. URL https://doi.org/10.1145/2882782.

[63] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM, 2020. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

[64] Weronika Lajewska and Krisztian Balog. From baseline to top performer: A reproducibility study of approaches at the TREC 2021 conversational assistance track. In Jaap Kamps, Lorraine Goeuriot, Fabio Crestani, Maria Maistro, Hideo Joho, Brian Davis, Cathal Gurrin, Udo Kruschwitz, and Annalina Caputo, editors, *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part III*, volume 13982 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2023. doi: 10.1007/978-3-031-28241-6_12. URL https://doi.org/10.1007/978-3-031-28241-6_12.

[65] Jimmy Lin. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 52(2):40–51, 2018. doi: 10.1145/3308774.3308781. URL https://doi.org/10.1145/3308774.3308781.

[66] Jimmy Lin. Building a culture of reproducibility in academic research. *CoRR*, abs/2212.13534, 2022. doi: 10.48550/arXiv.2212.13534. URL https://doi.org/10.48550/arXiv.2212.13534.

[67] Jimmy Lin and Qian Zhang. Reproducibility is a process, not an achievement: The replicability of IR reproducibility experiments. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, volume 12036 of *Lecture Notes in Computer Science*, pages 43–49. Springer, 2020. doi: 10.1007/978-3-030-45442-5_6. URL https://doi.org/10.1007/978-3-030-45442-5_6.

[68] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig MacDonald, and Sebastiano Vigna. Toward reproducible baselines: The open-source IR reproducibility challenge. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, volume 9626 of *Lecture Notes in Computer Science*, pages 408–420. Springer, 2016. doi: 10.1007/978-3-319-30671-1_30. URL https://doi.org/10.1007/978-3-319-30671-1_30.

[69] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Frassetto Nogueira. Pyserini: A python toolkit

for reproducible information retrieval research with sparse and dense representations. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2356–2362. ACM, 2021. doi: 10.1145/3404835.3463238. URL `https://doi.org/10.1145/3404835.3463238`.

[70] Jimmy Lin, Daniel Campos, Nick Craswell, Bhaskar Mitra, and Emine Yilmaz. Fostering coopetition while plugging leaks: The design and implementation of the MS MARCO leaderboards. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, editors, *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2939–2948. ACM, 2022. doi: 10.1145/3477495.3531725. URL `https://doi.org/10.1145/3477495.3531725`.

[71] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. Simplified data wrangling with ir$_datasets. In SIGIR$, 2021.

[72] Craig Macdonald and Nicola Tonellotto. Declarative experimentation ininformation retrieval using pyterrier. In *Proceedings of ICTIR 2020*, 2020.

[73] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016. URL `http://arxiv.org/abs/1611.09268`.

[74] Iadh Ounis, Giambattista Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Christina Lioma. Terrier: A high performance and scalable information retrieval platform. 01 2006.

[75] Joao Palotti, Harrisen Scells, and Guido Zuccon. Trectools: an open-source python library for information retrieval practitioners involved in trec-like campaigns. SIGIR'19. ACM, 2019.

[76] João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. A large-scale study about quality and reproducibility of jupyter notebooks. In Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc, editors, *Proceedings of the 16th International Conference on Mining Software*

Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada, pages 507–517. IEEE / ACM, 2019. doi: 10.1109/MSR.2019.00077. URL `https://doi.org/10.1109/MSR.2019.00077`.

[77] Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. TIRA integrated research architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*, volume 41 of *The Information Retrieval Series*, pages 123–160. Springer, 2019. doi: 10.1007/978-3-030-22948-1_5. URL `https://doi.org/10.1007/978-3-030-22948-1_5`.

[78] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models, 2021.

[79] Kirk Roberts, Dina Demner-Fushman, Ellen Voorhees, William R. Hersh, Steven Bedrick, Alexander J. Lazar, and Shubham Pant. Overview of the trec 2017 precision medicine track. In *TREC*, 2017.

[80] Kirk Roberts, Dina Demner-Fushman, Ellen Voorhees, William R. Hersh, Steven Bedrick, and Alexander J. Lazar. Overview of the trec 2018 precision medicine track. In *TREC*, 2018.

[81] Birgit Schmidt-Wesche, Robert Mack, Christian Lenz Cesar, and David VanEsselstyn. IBM search UI prototype evaluation at the interactive track of TREC-6. In Ellen M. Voorhees and Donna K. Harman, editors, *Proceedings of The Sixth Text REtrieval Conference, TREC 1997, Gaithersburg, Maryland, USA, November 19-21, 1997*, volume 500-240 of *NIST Special Publication*, pages 517–534. National Institute of Standards and Technology (NIST), 1997. URL `http://trec.nist.gov/pubs/trec6/papers/ibm-interactive.ps`.

[82] D. Sculley, Jasper Snoek, Alexander B. Wiltschko, and Ali Rahimi. Winner's curse? on pace, progress, and empirical rigor. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=rJWF0Fywf`.

[83] Trevor Strohman, Donald Metzler, Howard Turtle, and W. Croft. Indri: A language-model based search engine for complex queries. *Information Retrieval - IR*, 01 2005.

[84] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In Joaquin Vanschoren

and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html.

[85] Christophe Van Gysel and Maarten de Rijke. Pytrec_eval: An extremely fast python interface to trec_eval. In *SIGIR*. ACM, 2018.

[86] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[87] Ellen M. Voorhees. Nist trec disks 4 and 5: Retrieval test collections document set, 1996.

[88] Ellen M. Voorhees. The yangphy of information retrieval evaluation. In Carol Peters, Martin Braschler, Julio Gonzalo, and Michael Kluck, editors, *Evaluation of Cross-Language Information Retrieval Systems, Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001, Darmstadt, Germany, September 3-4, 2001, Revised Papers*, volume 2406 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2001. doi: 10.1007/3-540-45691-0_34. URL https://doi.org/10.1007/3-540-45691-0_34.

[89] Ellen M. Voorhees. Overview of the TREC 2004 robust track. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2004. URL http://trec.nist.gov/pubs/trec13/papers/ROBUST.OVERVIEW.pdf.

[90] Ellen M. Voorhees. The evolution of cranfield. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*, volume 41 of *The Information Retrieval Series*, pages 45–69. Springer, 2019. doi: 10.1007/978-3-030-22948-1_2. URL https://doi.org/10.1007/978-3-030-22948-1_2.

[91] Ellen M. Voorhees and Donna K. Harman. Overview of the seventh text retrieval conference (trec-7) [on-line]. 1999. URL `https://api.semanticscholar.org/CorpusID:13102016`.

[92] Ellen M. Voorhees and Donna K. Harman. Overview of the eighth text retrieval conference (trec-8). In *Text Retrieval Conference*, 1999. URL `https://api.semanticscholar.org/CorpusID:2540550`.

[93] Ellen M. Voorhees, Shahzad Rajput, and Ian Soboroff. Promoting repeatability through open runs. In Emine Yilmaz and Charles L. A. Clarke, editors, *Proceedings of the Seventh International Workshop on Evaluating Information Access, EVIA 2016, a Satellite Workshop of the NTCIR-12 Conference, National Center of Sciences, Tokyo, Japan, june 7, 2016*. National Institute of Informatics (NII), 2016. URL `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/evia/04-EVIA2016-VoorheesE.pdf`.

[94] Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. TREC-COVID: constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1):1:1–1:12, 2020. doi: 10.1145/3451964.3451965. URL `https://doi.org/10.1145/3451964.3451965`.

[95] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. CORD-19: the covid-19 open research dataset. *CoRR*, abs/2004.10706, 2020. URL `https://arxiv.org/abs/2004.10706`.

[96] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White, editors, *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 1253–1256. ACM, 2017. doi: 10.1145/3077136.3080721. URL `https://doi.org/10.1145/3077136.3080721`.

[97] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using lucene. *ACM J. Data Inf. Qual.*, 10(4):16:1–16:20, 2018. doi: 10.1145/3239571. URL `https://doi.org/10.1145/3239571`.

[98] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from

neural ranking models. In Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer, editors, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1129–1132. ACM, 2019. doi: 10.1145/3331184.3331340. URL `https://doi.org/10.1145/3331184.3331340`.