Supporting Interoperability Between Open-Source Search Engines with the Common Index File Format

Jimmy Lin,[1] Joel Mackenzie,[2] Chris Kamphuis,[3] Craig Macdonald,[4] Antonio Mallia,[5]
Michał Siedlaczek,[5] Andrew Trotman,[6] Arjen de Vries[3]

[1] University of Waterloo [2] The University of Melbourne [3] Radboud University
[4] University of Glasgow [5] New York University [6] University of Otago

ABSTRACT

There exists a natural tension between encouraging a diverse ecosystem of open-source search engines and supporting fair, replicable comparisons across those systems. To balance these two goals, we examine two approaches to providing interoperability between the inverted indexes of several systems. The first takes advantage of internal abstractions around index structures and building wrappers that allow one system to directly read the indexes of another. The second involves sharing indexes across systems via a data exchange specification that we have developed, called the Common Index File Format (Ciff). We demonstrate the first approach with the Java systems Anserini and Terrier, and the second approach with Anserini, JASSv2, OldDog, PISA, and Terrier. Together, these systems provide a wide range of implementations and features, with different research goals. Overall, we recommend Ciff as a low-effort approach to support independent innovation while enabling the types of fair evaluations that are critical for driving the field forward.

ACM Reference Format:
Jimmy Lin, Joel Mackenzie, Chris Kamphuis, Craig Macdonald, Antonio Mallia, Michał Siedlaczek, Andrew Trotman, and Arjen de Vries. 2020. Supporting Interoperability Between Open-Source Search Engines with the Common Index File Format. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401404

1 INTRODUCTION

Academic information retrieval researchers often share their innovations in open-source search engines, a tradition that dates back to the SMART system in the mid 1980s [2]. Today, there exists a vibrant ecosystem of IR toolkits capturing a variety of ranking models, query evaluation techniques, and other research innovations. Yet, as several replicability and reproducibility efforts have shown, it is often difficult to compare different systems in a fair manner, both in terms of retrieval effectiveness and query evaluation efficiency, on standard test collections [3, 7]. In terms of effectiveness, many mundane details such as the stemmer, stopwords list, and other difficult-to-document implementation choices matter a great deal, often having a greater impact than more substantive differences such as ranking models. These issues also affect efficiency-focused studies—for example, the presence or absence of stopwords alters skipping behavior during postings traversal.

On the one hand, a vibrant intellectual community demands diversity in terms of the tools available to researchers. On the other hand, the ability to conduct meaningful evaluations across systems is critical to driving progress. How can we meaningfully balance these two desiderata? Despite explorations of alternative formulations of keyword search [1], inverted indexes and associated data structures remain at the heart of nearly all IR systems today. Thus, if we are able to devise a mechanism for different search engines to share index structures, this would represent substantial progress towards achieving our aforementioned goals.

In principle, there are two ways such sharing can be accomplished: Since most search engine implementations have internal abstractions of index structures—providing support for basic operations such as postings lookup and traversal—it may be possible for one search engine to directly read the index structures created by another through an intermediate adaptor or wrapper. Alternatively, we could define a data exchange format through which one system exports its index, to be imported by another system. For expository convenience, we refer to the first as the "wrapper" approach and the second as the "data exchange" approach.

There are advantages and disadvantages to both approaches. The wrapper approach is only possible if the search engine implementations provide the necessary internal abstractions and that their definitions are (reasonably) aligned; feasibility is further constrained by technical practicalities. For example, interoperability might be possible between two JVM-based systems, but bridging a Java and a C++ implementation might be too onerous. Furthermore, this approach requires $n \times (n-1)$ distinct wrappers to support interoperability between n systems, as every system would need to wrap the index structures of every other system. A final disadvantage is the overhead involved in these wrappers, which might make fair efficiency-focused evaluations difficult to conduct.

A data exchange approach presents a different set of tradeoffs. As such structures are not meant to be operated on directly, each system would need to read the data and rewrite the indexes into the system's native representation, necessitating an extra step to enable interoperability. In order for the format to be general and robust, it is likely to be more verbose than each engine's native encoding, and thus this approach has the disadvantage of requiring researchers to distribute (across the network) large files that may be unwieldy to manipulate. On the plus side, though, this approach avoids quadratic interactions, as each system would only need to write an exporter and an importer of the exchange format

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401404

to support full interoperability [4]. Finally, data exchange incurs no performance penalty at query time, and thus can support fair efficiency evaluations.

This work demonstrates both approaches. First, we apply the wrapper approach to bridge Terrier and Anserini, both Java-based systems. Second, we propose a Common Index File Format (Ciff) and have built an index exporter that converts Lucene indexes into this format. Additionally, we have implemented importers to take Ciff and transform the data into the native representations of four other systems (JASSv2, OldDog, PISA, and Terrier), demonstrating interoperability by data exchange in practice.

After presenting experimental results using both approaches, we recommend the second and our Common Index File Format (Ciff) as the preferred method to enable rapid, decoupled independent research and exploration of ideas while enabling fair comparisons between systems that are critical to advancing the field.

2 EXPERIMENTAL SETUP

Our efforts brought together researchers who have built a number of open-source search engines (listed alphabetically by system):

- **Anserini** [15] is an IR toolkit built on the popular open-source Lucene search library.
- **JASSv2** [12], written in C++, uses an impact-ordered index and processes postings Score-at-a-Time. It can index TREC collections directly, but imports web collection indexes from ATIRE.
- **PISA** [10] is an efficiency-focused search system, containing many state-of-the-art indexing and retrieval techniques. PISA primarily uses document-ordered indexes and Document-at-a-Time query evaluation.
- **OldDog** [5] is an IR engine built using a relational database, named after the work of Mühleisen et al. [11]. Its design supports rapid prototyping through formulation of different SQL queries.
- **Terrier** [9] is an IR toolkit, first released in 2004. It is written in Java, and supports a large number of TREC collections and retrieval approaches, including BM25 and learning to rank.

For our experiments, we used the following two test collections:

- **`Robust04`:** TREC Disks 4 & 5, excluding Congressional Record, with TREC topics 301–450, 601-700.
- **`ClueWeb12B`:** The ClueWeb12-B13 web crawl from Carnegie Mellon University, with TREC topics 201–300.

The first is perhaps the most widely used test collection in IR. The goal of using `ClueWeb12B` is to demonstrate the scalability of our approach—to provide a sense of how large Ciff can get, and to confirm that these structures can still be manipulated on modern hardware with reasonable ease.

3 WRAPPERS

As an example of the wrapper approach, we describe how interoperability between Terrier and Anserini, both Java-based systems, is achieved by wrapping the Lucene indexes generated by Anserini in Terrier APIs, such that Terrier can directly traverse Lucene postings for query evaluation.

The Terrier wrapper[1] we have implemented for the Lucene `IndexReader` API allows a Terrier postings list iterator to directly call the underlying Lucene methods; the change is entirely transparent to Terrier. This works well for simple frequency-based and positional representations, but we did not implement index fields due to differences in how they are defined in the two systems. Results on `Robust04` are shown in Table 1, where it is now possible to compare different query expansion methods using essentially the same index. We note that differences in BM25 effectiveness are very small, while the various query expansion methods have at most 2% AP difference.

System	AP	P@30
Anserini (BM25)	0.2531	0.3102
Anserini (BM25+RM3)	0.2903	0.3365
Anserini (BM25+Axiomatic QE)	0.2896	0.3333
Terrier (BM25)	0.2530	0.3106
Terrier (BM25+Bo1 QE)	0.2931	0.3406
Terrier (BM25+RM3)	0.2945	0.3371
Terrier-Lucene (BM25)	0.2524	0.3091
Terrier-Lucene (BM25+Bo1 QE)	0.2890	0.3356
Terrier-Lucene (BM25+RM3)	0.2887	0.3284

Table 1: Comparison of Anserini, Terrier, and the Terrier wrapper for Anserini's Lucene indexes on `Robust04`.

Despite the feasibility of this approach here, we felt that the effort involved was too substantial to be scaled to more systems. In particular, since Terrier and Anserini were both implemented in Java, API-level integration was not too onerous. However, bridging either with, for example, a system implemented in C++ such as PISA or JASSv2, would involve much more effort. This motivated us to explore the data exchange approach more thoroughly.

4 COMMON INDEX FILE FORMAT

Our second approach to supporting interoperability among different search engines is to define a data exchange specification that we call the Common Index File Format (Ciff) whereby systems can share their inverted indexes and other associated data structures that are required for ranking. Critically, we intend for this to be an *exchange* format and not an *operational* one—that is, we expect each system to read Ciff and transform the contents into the system's own internal representation.

At a high level, Ciff defines a specification for serializing postings lists and other associated data structures necessary for search engines. Put into practice, the simplest workable exchange format could be based on plain text files. Postings lists have regular, repeating structure, and in principle, it would be possible to define a delimited text format for capturing these structures. However, we decided against this approach for several reasons: In such a scheme, metadata such as the semantics of the delimiters would need to be documented separately, and thus easily "lost". Additionally, there is no easy way to enforce the integrity and validity of a particular export—unless we explicitly build in error checks, in which case the format becomes even more complicated.

Ultimately, for serialization we decided to use Protocol Buffers (protobufs), which provide a language-neutral, platform-neutral extensible mechanism for serializing structured data that is widely

[1] https://github.com/cmacdonald/terrier-lucene

deployed in industry. Protobufs share some similarities with C `structs` in providing a language to define abstract data types that can be arbitrarily nested and repeated to represent lists. The protobuf specification restricts types to those found on nearly all platforms (e.g., 32-bit integers) and from a definition, the protobuf compiler can automatically generate code for reading and writing data in the specified format, supporting a multitude of languages and platforms.

The complete specification of the protobuf messages defined in Ciff are available at http://ciff.osirrc.io/. Lacking sufficient space to include here, we instead present a high-level overview: A Ciff export is comprised of a single, possibly compressed, file with a sequence of delimited protobuf messages, exactly as follows:

- a `Header` message, followed by
- exactly the number of `PostingsList` messages specified in the `num_postings_lists` field of the `Header`, followed by
- exactly the number of `DocRecord` messages specified in the `num_docs` field of the `Header`.

A Ciff export begins with a `Header` that captures metadata such as versioning information, global index statistics, and a description of how the export was generated. The `num_postings_lists` field specifies the number of postings lists that are included in a particular export, which allows Ciff to support the use case of including only postings lists that correspond to a particular set of evaluation topics. Naturally, such a setting yields an export that is far smaller than the export of a complete index. A `PostingList` contains the term, its document frequency, its collection frequency, and a number of individual `Posting` messages equal to the document frequency. Following standard conventions, the `docid` is encoded as gaps. A Ciff export ends with document-specific information that is captured by a series of `DocRecord` messages, which contain the integer docid (referenced in the postings lists), the external collection docid (a string), and the length of the document.

A reference implementation that generates (and reads) Ciff exports from Lucene indexes built by Anserini is open-source and available at http://ciff.osirrc.io/. We have also implemented importers for all the other systems described in Section 2; links to code can also be found in our repository. The complete Anserini Lucene Ciff exports of the `Robust04` and `ClueWeb12B` indexes used in our experiments are 162 MiB and 25 GiB, respectively, as gzipped files; exports that contain only the query terms are 17 MiB and 1.3 GiB (compressed), respectively. Links to all these exports can be found at the above URL as well. We see that, even for reasonably large web collections that are used in IR research, Ciff exports are modest in size for modern hardware, both to ship across the network and to manipulate on disk.

4.1 Case Study: BM25 Variants

With Ciff, it is possible to conduct meaningful evaluations of ranking models from diverse systems that completely factor out the effects of different document processing pipelines (i.e., document cleaning regimes, tokenization, stopwords, etc.). We illustrate with a simple case study examining "BM25".

One major finding from previous replicability studies [3, 7] is that systems purporting to implement BM25 can exhibit large effectiveness differences on standard test collections. This is due to a combination of two factors: First, systems have different document processing pipelines; details like data cleaning make a big difference, but are relatively uninteresting to researchers. Second, "BM25" actually encompasses a large number of variants. However, Trotman et al. [14] and Kamphuis et al. [6] found that such differences are unlikely to be statistically significant. In both cases, this conclusion was arrived at by the authors implementing *all* the variants in *the same search engine* to support the comparisons. Needless to say, this is a time-consuming task, and not scalable in the general case, where we would like to compare *arbitrary* ranking functions from *any* search engine. This is exactly where Ciff comes in: With our exchange format, it is possible to conduct fair evaluations of ranking effectiveness on *different* systems.

System	`Robust04`		`ClueWeb12B`	
	AP	P@30	NDCG	ERR
Native Document Processing				
JASSv2	0.2570	0.3157	0.1132	0.0809
PISA	0.2543	0.3139	0.1169	0.0845
Terrier	0.2530	0.3106	0.1308	0.0978
Anserini	0.2531	0.3102	0.1340	0.0970
Common Index File Format				
JASSv2	0.2524	0.3096	0.1311	0.0937
PISA	0.2519	0.3083	0.1345	0.0971
OldDog-A	0.2531	0.3102	0.1345	0.0971
OldDog-L	0.2530	0.3102	0.1345	0.0971
Terrier	0.2524	0.3091	0.1321	0.0956

Table 2: Comparison of BM25 variants.

To illustrate, we present a simple, multi-system study of BM25 variants. For `Robust04` and `ClueWeb12B`, we exported the Lucene indexes generated by Anserini into Ciff (see previous section), which is then imported by all the remaining systems. We evaluated each system's BM25 ranking using standard metrics: AP (at rank 1000) and P@30 for `Robust04`, NDCG@10 and ERR@10 for `ClueWeb12B`. In all cases we set $k_1 = 0.9$ and $b = 0.4$, per the recommendations of Trotman et al. [13]. These results are shown in Table 2. In the top block of the table, we present figures from each system's "native" document processing pipeline to provide points of reference. Note that since Anserini is built directly on Lucene, its Ciff and "native" results are identical. For OldDog, we report both ATIRE BM25 (OldDog-A) and Lucene BM25 (OldDog-L).

There are three sources of differences in systems' rankings: (1) implementation of the document processing pipeline, (2) variants of the BM25 scoring function (including different parameter settings, quantization effects when computing impact scores, etc.), and (3) tie-breaking effects. With Ciff, we have eliminated the first effect. The third effect has been characterized in previous work [8, 16] and is mitigated here because Ciff ensures that documents are consistently ordered across all systems. Thus, this experiment allows us to isolate the effects of BM25 variants, although we must still manually ensure that every system uses the same parameter settings. In short, we have replicated previous replicability studies [6, 14], but in a manner that supports *cross-system* comparisons.

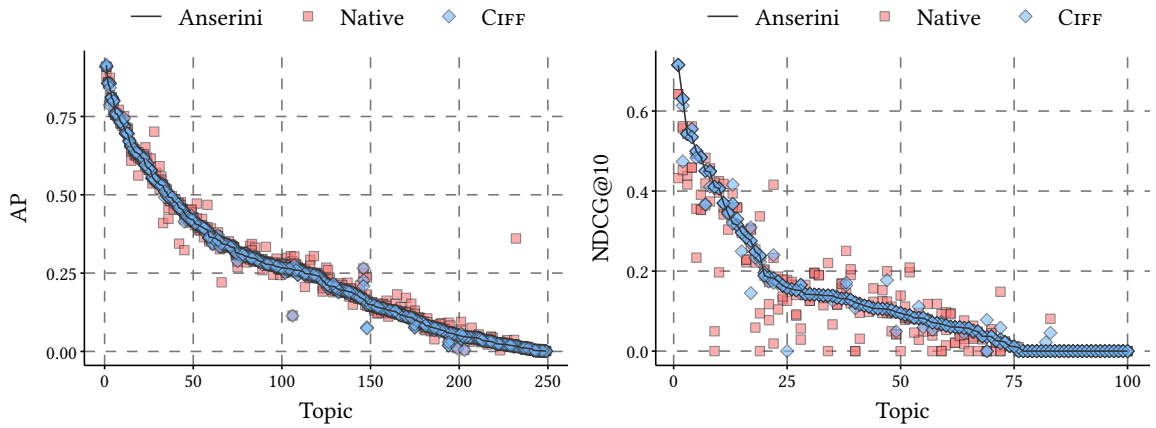

Figure 1: Per-topic scores for all systems, sorted in descending order of the metric (y-axis), based on Anserini's scores: `Robust04` on left and `ClueWeb12B` on right.

From Table 2, we see that effectiveness differences between the various systems with native document processing are larger than with Ciff. This effect is particularly noticeable with `ClueWeb12B`: On web documents, document processing (e.g., cleaning of HTML) has a much larger impact on effectiveness compared to `Robust04`, which comprises relatively clean SGML documents.

We conducted a Tukey's HSD (honestly significant difference) test for all the "native" systems as a group and with Ciff as a group: None of the differences are statistically significant, for both `Robust04` and `ClueWeb12B`. Nevertheless, if we examine per-topic scores, the differences between each system's native document processing pipeline and Ciff become much more prominent. Consider Figure 1 (left), which plots the per-topic AP scores for Anserini on `Robust04`, in decreasing order of effectiveness. We have overlaid the scores for the corresponding topics from all systems for both the native and Ciff conditions. Clearly, we see that Ciff reduces most of the per-topic effectiveness differences between systems. This experiment was repeated on the `ClueWeb12B` collection, using NDCG@10 as the metric; results are shown in Figure 1 (right). Once again, although there remain differences between systems' scores under Ciff, the conflating issue of document processing has been eliminated, thereby allowing researchers to more accurately characterize effectiveness.

Our simple case study demonstrates how Ciff supports meaningful cross-system comparisons. This approach can be easily extended to evaluations of different ranking models, candidate generation techniques in multi-stage ranking pipelines, performance comparisons of query latency, and beyond.

5 CONCLUSIONS

We envision Ciff to be an ongoing, open, and community-driven effort that allows researchers to independently pursue their own lines of inquiry while supporting fair and meaningful evaluations. Additional contributions are most welcome! As our efforts gain traction, we envision future research papers adopting "standard" Ciff exports in their experiments—this would have the dual benefit of standardizing empirical methodology and more clearly highlighting the impact of proposed innovations.

ACKNOWLEDGMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, Compute Ontario and Compute Canada, the Australian Research Council (ARC) Discovery Grant DP170102231, the US National Science Foundation (IIS-1718680), and research program Commit2Data with project number 628.011.001 financed by the Dutch Research Council (NWO).

REFERENCES

[1] Leonid Boytsov, David Novak, Yury Malkov, and Eric Nyberg. 2016. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. In *CIKM*.
[2] Chris Buckley. 1985. *Implementation of the SMART Information Retrieval System*. Department of Computer Science TR 85-686. Cornell University.
[3] Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu. 2019. Overview of the 2019 Open-Source IR Replicability Challenge (OSIRRC 2019). In *CEUR Workshop Proceedings Vol-2409*.
[4] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trotman. 2017. A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation. In *WSDM*.
[5] Chris Kamphuis and Arjen de Vries. 2019. The OldDog Docker Image for OSIRRC at SIGIR 2019. In *CEUR Workshop Proceedings Vol-2409*.
[6] Chris Kamphuis, Arjen de Vries, Leonid Boytsov, and Jimmy Lin. 2020. Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants. In *ECIR*.
[7] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig Macdonald, and Sebastiano Vigna. 2016. Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge. In *ECIR*.
[8] Jimmy Lin and Peilin Yang. 2019. The Impact of Score Ties on Repeatability in Document Ranking. In *SIGIR*.
[9] Craig Macdonald, Richard McCreadie, Rodrygo L.T. Santos, and Iadh Ounis. 2012. From puppy to maturity: Experiences in developing Terrier. *OSIR Workshop at SIGIR*.
[10] Antonio Mallia, Michał Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *CEUR Workshop Proceedings Vol-2409*.
[11] Hannes Mühleisen, Thaer Samar, Jimmy Lin, and Arjen de Vries. 2014. Old Dogs Are Great at New Tricks: Column Stores for IR Prototyping. In *SIGIR*.
[12] Andrew Trotman and Matt Crane. 2019. Micro- and Macro-optimizations of SaaT Search. *Software: Practice and Experience* 49, 5 (2019), 942–950.
[13] Andrew Trotman, Xiang-Fei Jia, and Matt Crane. 2012. Towards an Efficient and Effective Search Engine. In *SIGIR 2012 Workshop on Open Source Information Retrieval*.
[14] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *ADCS*.
[15] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *Journal of Data and Information Quality* 10, 4 (2018), Article 16.
[16] Ziying Yang, Alistair Moffat, and Andrew Turpin. 2016. How Precise Does Document Scoring Need to Be?. In *AIRS*.