A Lightweight Environment for Learning Experimental IR Research Practices

Zeynep Akkalyoncu Yilmaz, Charles L. A. Clarke, and Jimmy Lin
David R. Cheriton School of Computer Science, University of Waterloo
http://github.com/castorini/anserini-notebooks-afirm2020

ABSTRACT

Tools, computing environments, and datasets form the three critical ingredients for teaching and learning the practical aspects of experimental IR research. Assembling these ingredients can often be challenging, particularly in the context of short courses that cannot afford large startup costs. As an initial attempt to address these issues, we describe materials that we have developed for the "Introduction to IR" session at the ACM SIGIR/SIGKDD Africa Summer School on Machine Learning for Data Mining and Search (AFIRM 2020), which builds on three components: the open-source Lucene search library, cloud-based notebooks, and the MS MARCO dataset. We offer a self-reflective evaluation of our efforts and hope that our lessons shared can benefit future efforts.

ACM Reference Format:
Zeynep Akkalyoncu Yilmaz, Charles L. A. Clarke, and Jimmy Lin. 2020. A Lightweight Environment for Learning Experimental IR Research Practices. In *Proceedings of the 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/10.1145/3397271.3401395

1 INTRODUCTION

Teaching and learning the practical aspects of experimental IR research requires overcoming substantial barriers in terms of obtaining the appropriate tools, computing environments, and datasets. While there exists many open-source toolkits and libraries to support IR research, downloading, installing, and configuring them is often not an easy task. Students' computing environments (typically, personal laptops) are diverse in terms of specifications, operating systems, and configuration details. Shared teaching resources available in universities are more homogeneous, but are frequently restrictive in supporting custom configurations. Thus, simply getting the tools to run in a compatible computing environment can require Herculean efforts. For semester-long courses, the pain can be managed, but one-day tutorials and other short courses can scarcely afford large startup costs.

Beyond tools and computing environments, it is of course impossible to teach and learn about experimental IR research without test collections. Most research test collections, such as those built in the context of TREC, NTCIR, CLEF, FIRE, and other evaluation campaigns, are protected by usage agreements and thus cannot be freely downloaded. This restricts their usage in publicly-accessible teaching materials.

The barriers associated with tools, computing environments, and datasets can already be onerous in regions of the world that are relatively well-resourced. These challenges are further magnified in developing regions of the world, where the community would like to cultivate future information retrieval researchers.

We describe our initial attempts to tackle these challenges, built around three main components:

- The open-source Java-based Lucene search library, using the Python interface provided by the Anserini IR research toolkit.
- Cloud-based notebooks (specifically, Google Colab), for a broadly accessible, homogeneous computing environment.
- The MS MARCO dataset, which is not only publicly downloadable but also widely used in many recent evaluations.

The immediate impetus for the creation of the resources described in this paper was the ACM SIGIR/SIGKDD Africa Summer School on Machine Learning for Data Mining and Search (AFIRM 2020), held in Cape Town, South Africa in January 2020.[1] This summer school was the second in a series of annual events intended to support new faculty and advanced graduate students in growing an African research community focused on information retrieval and data mining. The resources described in this paper were deployed in the "Introduction to IR" session. Although created for a specific purpose, we believe that our technical approach (i.e., the combination of tools, environments, and datasets) as well as our materials can be adapted to other learning contexts. Thus, we share what we have built with the community to solicit feedback. We offer a self-reflective assessment based on our AFIRM offering, and hope that others can improve upon our efforts in the future.

2 COMPONENTS

This section describes our technical approach to tools, computing environments, and datasets for AFIRM. We note that, individually, these components are not novel, but we argue that only recently has this triumvirate been assembled into a cohesive and convenient package to support learning experimental IR research practices.

2.1 Lucene

While there are many open-source search engines [2, 6], Lucene stands alone as the only one to have achieved broad deployment in industry for practical applications. In fact, for IR practitioners building real-world search solutions, Lucene has become the *de facto*

[1]http://sigir.org/afirm2020/

standard, its capabilities typically accessed through Elasticsearch or Solr, higher-level platforms using Lucene in their cores.

Teaching experimental IR research using Lucene has the nice side effect of providing a marketable skill for students who do not end up pursuing advanced degrees but primarily desire an industry position as a practitioner. However, out of the box, Lucene is ill-suited for IR research because it provides no obvious entry point for running experiments on standard IR test collections. The Anserini IR toolkit[2] [10, 11] addresses this issue: it builds around Lucene the facilities to support standard IR evaluation methodologies.

However, even with Anserini, we have noticed a gap between what is offered and what is desired. Over the past few years, Python has become the programming language of choice for data scientists and other applied machine learning practitioners, particularly those training neural networks. The fact that Lucene is written in Java presents a hurdle to adoption for many.

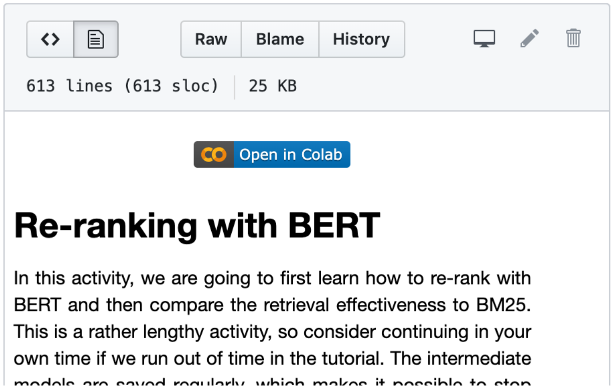
The Anserini team recently addressed this gap by building Pyserini, which is a Python wrapper around Anserini. Pyserini provides a self-contained package available in the PyPI (Python Package Index) repository,[3] such that a simple `pip install` command is sufficient to download, install, and configure Anserini on a standard Python 3.6 distribution. This provides a bridge between Python and Lucene, running on an embedded JVM in a manner that is completely transparent to the developer. Pyserini, Anserini, and Lucene form the toolchain that our AFIRM materials build around.

2.2 Notebooks

Ideally, we desire a computing environment that appears fully configured with the relevant tools installed and operational at the click of a mouse. With cloud-based notebooks in general (the most popular of which is Jupyter[4]), and Google's Colab hosted notebook service in particular, we come quite close to exactly such an experience. Many core features of notebooks, such as the interleaving of code and narrative prose, date back to the concept of literate programming from the 1980s. However, it wasn't until a few years ago that notebooks have become an indispensable part of data scientists' arsenals; for many, they have replaced the command-line shell as the execution environment of choice. One can easily find on the web myriad tutorials written as notebooks, on topics ranging from introductory programming concepts to advanced neural network modeling. Discussions of notebooks are found in the IR literature as well [3, 7], and in fact, Guido Zuccon provided notebook adaptations of his lesson plans for AFIRM 2019.[5]

The materials that we have developed for AFIRM 2020 take the form of Python notebooks stored in a GitHub repository, such that with a single mouse click (see Figure 1), the contents are loaded directly into Google's Colab environment. An interactive notebook springs forth, ready to replicate demonstration material or accept students' solutions to exercises that are interspersed in the contents (more details in the next section). Each notebook begins with all the invocations necessary to set up the execution environment, for example, installation of various dependent software packages. The student simply needs to click "run" to produce the desired output.

[2]http://anserini.io
[3]https://pypi.org/project/pyserini
[4]https://jupyter.org
[5]https://github.com/ielab/afirm2019



Re-ranking with BERT

In this activity, we are going to first learn how to re-rank with BERT and then compare the retrieval effectiveness to BM25. This is a rather lengthy activity, so consider continuing in your own time if we run out of time in the tutorial. The intermediate

Figure 1: Screenshot showing the start of the re-ranking notebook, with the link to launch it in Google Colab. We needed to individually assist many students with the process of launching Colab, suggesting that information on Colab itself should be included in the notebooks.

The exact same experience can be replicated for any student with a browser, anywhere in the world. While Google's service is useful (especially since it is currently free to use and exploits Google's vast cloud infrastructure, thus offering large-scale storage and computing solutions), in principle we could have managed our own Jupyter notebook server to achieve largely the same effect. Under this mode of "delivering" computing environments, the server only needs to be installed and configured once (and thus represents a tolerable investment of effort).

2.3 MS MARCO

Of course, tools and computing environments are useless unless there are documents to index, queries to execute, and relevance judgments to compute metrics with. Standard research test collections, such as those from TREC, cannot be publicly distributed since users must first sign usage agreements. This makes building teaching materials around them cumbersome.

With the introduction of the MS MARCO dataset [1], the community now has access to test collections that are publicly accessible—by that, we mean content residing at public URLs that can be downloaded freely (at least for our purposes) without needing to sign any usage agreements. In fact, the setup portions of our notebooks for AFIRM include fetching data from these sources with `wget`. To be clear, the data transfer occurs between the server hosting the data and the notebook server, *not* the client browser. The MS MARCO dataset actually comprises several test collection; we focus on the passage collection, which has been used for several tracks at TREC 2019, including the Deep Learning Track [4] and the Conversational Assistance Track [5]. As such, it represents a realistic collection for learning the basics of IR research.

3 OVERVIEW OF THE NOTEBOOKS

AFIRM 2020 offered multiple all-day sessions on different facets of information retrieval and data mining, each of which included morning lectures followed by hands-on labs in the afternoons. Our "Introduction to IR" was the first offered at the event and opened with a lecture on the topic. The notebooks presented here were used in this introductory session. All materials may be found at the following URL:

http://github.com/castorini/anserini-notebooks-afirm2020

While this paper specifically describes the version used at AFIRM 2020, we continue to evolve the notebooks. Before the summer school, the notebooks were tested by Waterloo students new to the field of information retrieval, and we may continue to use the notebooks for onboarding students into our research group. We hope these notebooks will become a more widely used resource for IR education.

As an explicit requirement, the notebooks can be run in any modern browser with minimal external bandwidth. Indeed, they ran successfully in regular computer science teaching labs at the University of Cape Town, as well as in various browsers running on students' personal devices. At the time of the event, there were no known Google data centers in Africa,[6] but only minor technical problems were encountered during transfer of large data.

We intend for these notebooks to be self-explanatory; students should be able to complete them on their own with little or no help. If this were already the case in practice, we could merely publicize these materials at SIGIR and other venues, ready to be widely deployed. Unfortunately, from our experience at AFIRM 2020, work remains to be done (see Section 4). We hope to obtain feedback from the broader research community to aid us in evolving the notebooks into a generally valuable resource.

We structured the AFIRM 2020 lab into four notebooks, which build on each other but can be completed independently, depending on past knowledge: (1) Python for Information Retrieval, (2) Indexing, (3) Ranking, and (4) Re-ranking.

Material in these notebooks build on concepts introduced during the morning lectures, where the basics of indexing, ranking, re-ranking, and evaluation were introduced. The morning lectures also discussed commercial search, including e-commerce search, social media search, music search, and recommender systems, as well as traditional web search. Emphasis was placed on the need for scalability and on the practical difficulties of evaluating search engines. The basics of online and offline evaluation were covered. Problems interpreting implicit feedback signals were highlighted. As a result, by the afternoon lab, the notebooks were properly situated in their technological and scientific contexts.

Python for Information Retrieval. This first notebook assumes a basic knowledge of Python and the Unix shell, and provides foundational exercises for students who are completely new to IR. After introducing some basic features of the notebooks, including the execution of shell commands, we step through standard Python text processing operations, which also serve as a gentle introduction to Python. These operations include converting text to lowercase and performing simple tokenization.

This notebook culminates in an end-to-end exercise to build an inverted index for a small collection of passages using standard Python data structures. We suggest that students create a dictionary mapping tokens to lists of pairs, where each pair contains a document identifier and a term frequency value. Our goal is to provide a basic understanding of the indexing structures created using the tools in the second notebook.

Indexing. The second notebook starts by walking the student through the process of installing the required tools and packages, including Java, Maven, and Anserini, and downloading all required data, including the MS MARCO passage collection, queries, and relevance judgments. The entire process is as simple as re-running the cells in the notebook. Students explore the data with `grep` and other Unix commands to establish familiarity with the format and content of the data. We highlight the relationship between passages, queries, and relevance judgments.

We then introduce Anserini's indexing command, with an explanation of possible options. Students build an index and explore it with Pyserini, Anserini's Python interface. Using Pyserini, students can compute document and term frequencies, display postings lists, generate document vectors, view raw passages, and directly compute ranking scores. Our goal is to illustrate relationships between the dictionary-based inverted index from the first notebook and the index structures created by Anserini.

Ranking. In the third notebook, students interactively query the MS MARCO collection with built-in ranking functions and generate TREC runs. An example query ("south african football teams") illustrates the impact of BM25 parameter changes and pseudo-relevance feedback. Students are encouraged to create their own example queries and to explore the impact of those techniques.

Students then execute a full query set over the collection in batch mode. We introduce `trec_eval`, including file formats and options. Students apply `trec_eval` to their runs, illustrating on a larger scale the impact of parameter changes and pseudo-relevance feedback. At this point, they are free to experiment further or to continue on to the re-ranking notebook.

Re-ranking. No tutorial today would be complete, of course, without BERT. The lab concludes with a re-ranking notebook that provides an example of modern neural methods for search based on Nogueira and Cho [9]. At AFIRM 2020, BERT was briefly introduced during the morning lecture, and this notebook provides direct experience. Lectures later in the summer school explored neural methods in greater depth.

Students re-rank the output generated from the previous notebook with BERT and compare retrieval effectiveness to BM25. They start by setting up the Colab TPU environment, verifying that a TPU device is successfully connected and uploading credentials to the TPU for GCS bucket usage. After some data transform steps, they fine-tune BERT for passage re-ranking, re-rank the passages, and evaluate the result. Since this notebook requires a working knowledge of Tensorflow,[7] a machine learning library widely adopted in both industry and research communities, it is left as an optional exercise for the motivated students.

4 EXPERIENCE AND DISCUSSION

The diversity of backgrounds at AFIRM 2020 posed a challenge in developing a lab that would benefit all students. Although notebooks allow students to progress at their own pace, we found that

[6]https://www.google.com/about/datacenters/locations

[7]https://www.tensorflow.org

they were insufficient as standalone teaching tools. We also observed that students who worked collaboratively in small groups navigated the notebooks more effectively and developed a deeper understanding through discussions.

The extendable nature of Colab notebooks seemed to complement the lab exercises well. Not only did the notebooks allow the students to step through a complete basic IR experiment, but they also prompted students to write snippets of code from scratch to solidify their understanding of the underlying concepts. Given the proper tools in the form of Pyserini methods, students found it especially helpful to be able to explore the data structures under the hood, e.g., to interact with the index.

Since our goal is to create the best possible resource for teaching and learning the practice of IR research, we outline some shortcomings we already recognize. We hope to solicit additional feedback from the community on these and other issues.

Pedagogical Shortcomings. Markov and de Rijke [8] present a review of how information retrieval is currently taught by textbooks and tutorials. They identify a greater focus on *ad hoc* retrieval (vs. question answering or recommendation) and a greater focus on offline evaluation (vs. online evaluation). They argue for a more balanced curriculum, including conversational and mobile search, as well as interaction mining, query understanding, and online learning to rank. We do little to address these recommendations. Our notebooks cover exactly the traditional *ad hoc* retrieval scenario with offline evaluation that Markov and de Rijke identified as forming the core of current material. We do include a notebook on neural re-ranking and work with a collection of passages suitable for question answering, but do not address any of the other topics suggested by the authors.

Presentational Shortcomings. The notebooks as deployed for AFIRM have expectations with respect to background that may not be realistic for a range of potential students. The notebooks assume a basic familiarity with both Python and the Unix shell. The necessary background in Python programming is beyond what could be covered in the context of an IR tutorial. This background is perhaps best provided through any of the many available tutorials on basic Python programming, including those that are already in the form of notebooks. On the other hand, a basic explanation of Unix shell commands should have been incorporated into the notebook, especially since only the most basic of commands are required (`ls`, `wc`, `cat`, `grep` for a fixed string, etc.).

The re-ranking notebook, in particular, had some significant presentational shortcomings. We overlooked the definition of some key terms that would have promoted understanding. For example, the term "TPU" is used without explanation. The final re-ranking and evaluation steps are presented as a single chunk of Python code, which should have been broken down for clarity. These oversights might be explained by our relative lack of experience in teaching to more general audiences.

The notebooks also have a strong focus on English, generally ignoring text processing in non-English languages. For example, case normalization and stopword elimination are considered only in terms of English. Given our goal of making IR tools more generally accessible, it would be good to include pointers to resources outside English. At AFIRM, an invited talk immediately following the lab discussed low resource languages, and the lack of an explicit connection represented a missed opportunity.

Colab Shortcomings. At the time of writing, Google Colab provides a nearly ideal environment for our purposes, but we note a few minor issues that can create substantial friction. In particular, various popups and interstitial pages related to security can create uncertainty and alarm. Running any Colab notebook created by a third party generates a pop-up warning about potential risks, with no clear way for an inexperienced student to assess those risks. The re-ranking notebook generates an interstitial page indicating that "Google Cloud SDK requires access to your Google Account" and that access would allow it to "delete all of your Google Drive files". While these warnings are accurate, we wish there were ways to avoid them or to make the risks clearer. Ideally, notebooks could be given finer-grained access control permissions, since we only need to access students' Google Drives to save modified notebooks.

Missing Context. AFIRM 2020 students undertook the lab supported by the morning lectures, which placed the notebooks in the context of IR research. This context is missing from the notebooks, so as a resource for learning they still need classroom support. Ideally, the notebooks should evolve into a standalone resource.

5 CONCLUSION

Prior to the pandemic, we had planned to return to AFIRM 2021 with improved notebooks, reflecting feedback from the community. We still hope to return to AFIRM in the future. Independent of these specific materials, our technical approach of combining Lucene with notebooks applied to MS MARCO data presents a powerful combination that might be useful for other purposes.

ACKNOWLEDGMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We'd like to thank SIGIR/SIGKDD for sponsoring the AFIRM series and the students for their feedback.

REFERENCES

[1] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268* (2016).

[2] R. Clancy, N. Ferro, C. Hauff, J. Lin, T. Sakai, and Z. Wu. 2019. Overview of the 2019 Open-Source IR Replicability Challenge (OSIRRC 2019). In *CEUR Workshop Proceedings Vol-2409*. 1–7.

[3] R. Clancy, J. Lee, Z. Yilmaz, and J. Lin. 2019. Information Retrieval Meets Scalable Text Analytics: Solr Integration with Spark. In *SIGIR*. 1313–1316.

[4] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos. 2019. Overview of the TREC 2019 Deep Learning Track. In *TREC*.

[5] J. Dalton, C. Xiong, and J. Callan. 2019. CAsT 2019: The Conversational Assistance Track overview. In *TREC*.

[6] J. Lin, M. Crane, A. Trotman, J. Callan, I. Chattopadhyaya, J. Foley, G. Ingersoll, C. Macdonald, and S. Vigna. 2016. Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge. In *ECIR*. 408–420.

[7] C. Macdonald. 2018. Combining Terrier with Apache Spark to Create Agile Experimental Information Retrieval Pipelines. In *SIGIR*. 1309–1312.

[8] I. Markov and M. de Rijke. 2019. What Should We Teach in Information Retrieval? *SIGIR Forum* 52, 2 (January 2019), 19–39.

[9] R. Nogueira and K. Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).

[10] P. Yang, H. Fang, and J. Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *SIGIR*. 1253–1256.

[11] P. Yang, H. Fang, and J. Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *Journal of Data and Information Quality* 10, 4 (2018), Article 16.