H_2oloo at TREC 2018: Cross-Collection Relevance Transfer for the Common Core Track

Ruifan Yu, Yuhao Xie, and Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo
jimmylin@uwaterloo.ca

1 INTRODUCTION

The h2oloo team at the University of Waterloo participated in the Common Core Track in TREC 2018. Our main effort involved reproducing the cross-collection relevance transfer technique of Grossman and Cormack [8] from the TREC 2017 Common Core Track, as captured in their WCrobust0405 run. Their idea was relatively simple: for information needs (topics) that are shared across more than one test collection, it is possible to train (per topic) relevance classifiers using one or more test collections (in their case, from the TREC 2004 and 2005 Robust Tracks) and apply the classifiers to a new document collection (in their case, the New York Times collection used in the TREC 2017 Common Core Track) to improve ranking effectiveness. Each classifier, in essence, learns a relevance model for a specific information need, and the experiments of Grossman and Cormack demonstrate that such models can generalize across document collections.

According to the TREC 2017 Common Core Track overview paper [2], WCrobust0405 ranked third in terms of average precision of runs that contributed to the judgment pools. The two runs that were more effective than WCrobust0405 involved humans who interactively searched the target collection to find relevant documents. In other words, the relevance transfer technique yielded effectiveness levels that approach human searchers.

Not only is the technique of Grossman and Cormack effective, it is also simple: According to their paper, a logistic regression classifier for each topic was trained on the union of relevance judgments from the TREC 2004 and 2005 Robust Tracks. Documents were represented using word-level *tf-idf* features and each logistic regression classifier was learned using Sofia-ML[1] and then applied to the entire Common Core collection. The top 10000 scoring documents (per topic), in decreasing order of classifier score, was submitted as the final run.

We set out to reproduce the work of Grossman and Cormack described above, but with three main differences:

- *Reranking search results.* Instead of applying relevance classifiers over the *entire* collection, we reranked only the top $k = 10000$ hits from an initial retrieval run.
- *Incorporating retrieval scores.* In WCrobust0405, documents were simply sorted by their classifier scores. Since we were reranking an initial pool of candidate documents, it made sense to combine classifier scores with the retrieval scores, which we accomplished via linear interpolation.
- *Leveraging widely-used open-source tools.* The original Grossman and Cormack source code comprised a series of bash scripts that were not documented. For reusability and to support further explorations, we built our implementation on widely-used open-source tools: the Python machine learning package scikit-learn and the Anserini IR toolkit built on Lucene [11, 12].[2]

Our full reproduction effort is detailed in a forthcoming ECIR paper [14]. Our successful reimplementation was then applied to the Common Core Track in TREC 2018.

Overall, our work involved a collaboration with the Anserini team, who participated in the CENTRE, Common Core, and News Tracks separately. Although the team composition had some overlap, work on expanding the capabilities of Anserini involved researchers beyond the University of Waterloo. Thus, those efforts are documented in a separate TREC report [13].

2 RESULTS

We submitted a total of ten runs for the Common Core Track, based on the code developed for the reproduction effort described in Yu et al. [14]. Our reranking approach built on runs generated by Anserini. Of the 50 topics in the evaluation, 25 topics overlapped with topics from the TREC 2004 Robust Track (Robust04) and the TREC 2017 Common Core Track (Core17). Of those 25 topics, 15 overlapped with topics from the TREC 2005 Robust Track (Robust05). For these topics, we used all available relevance judgments from the previous test collections. For the remaining topics, we simply used the output of Anserini, unmodified.

The exact configurations of our submitted runs are shown in Table 1. There are three main degrees of freedom in our experimental design: First, we can vary the source of the candidate documents on which we apply our classifiers. This is shown in the column denoted "Base", where we use either BM25 with axiomatic semantic term matching [5, 10] or BM25 with RM3 [1]; additional details can be found in the Anserini overview paper [13]. In both cases, we retrieved the top $k = 10000$ documents from the collection. The second is the weight we place on the classifier score when integrating evidence with the retrieval score (via linear interpolation). This is shown in the column denoted "Weight". Details of weight tuning are described in Yu et al. [14].

The third degree of freedom is the actual classifier that we deployed, shown in Table 1 under the column "Classifier". We experimented with different classifiers and ensembles, but in all cases each classifier was trained using all available data from Robust04, Robust05, and Core17, where the feature vectors are terms with *tf-idf* weights. The feature space is shown under the column "Vocabulary": we tried using only the vocabulary of the Robust04 collection as well as the combined vocabulary of all three collections (All). For logistic regression, we actually experimented with two different

[1] https://github.com/glycerine/sofia-ml

[2] http://anserini.io/

	Run	Base	Weight	Classifier	Vocabulary	AP	NDCG	P@10	Pool
1	`h2oloo_LR2AX0.6`	BM25 + Ax	0.6	LR2	Robust04	0.3256	0.6145	0.5800	N
2	`h2oloo_LR2_rm3`	BM25 + RM3	0.6	LR2	Robust04	0.3273	0.6113	0.5800	N
3	`h2oloo_LRax0.6`	BM25 + Ax	0.6	LR2	All	0.3227	0.6123	0.5800	Y
4	`h2oloo_e7ax0.6`	BM25 + Ax	0.6	7 classifier ensemble	All	0.3310	0.6215	0.5840	Y
5	`h2oloo_e7ax0.7`	BM25 + Ax	0.7	7 classifier ensemble	All	0.3274	0.6209	0.5880	N
6	`h2oloo_e7rm30.6`	BM25 + RM3	0.6	7 classifier ensemble	All	0.3333	0.6143	0.5820	N
7	`h2oloo_e7rm30.7`	BM25 + RM3	0.7	7 classifier ensemble	All	0.3361	0.6177	**0.5940**	N
8	`h2oloo_enax0.6`	BM25 + Ax	0.6	3 classifier ensemble	All	0.3341	**0.6233**	0.5860	Y
9	`h2oloo_enax0.7`	BM25 + Ax	0.7	3 classifier ensemble	Robust04	0.3351	0.6218	0.5920	Y
10	`h2oloo_enrm30.6`	BM25 + RM3	0.6	3 classifier ensemble	All	**0.3382**	0.6193	0.5920	Y
	`anserini_bm25`	BM25	-	-	-	0.2284	0.5064	0.4500	Y
	`anserini_ax`	BM25 + Ax	-	-	-	0.2734	0.5582	0.4960	Y
	`anserini_rm3`	BM25 + RM3	-	-	-	0.2680	0.5422	0.4680	Y

Table 1: The configuration and effectiveness of our submitted runs; results from Anserini provided for reference.

configurations, what we call LR1 and LR2. The primary difference is that LR2 uses the so-called "balanced" mode in `scikit-learn` to automatically adjust class weights to be inversely proportional to class frequencies.

The first three rows of Table 1 describe submissions that used only one classifier (LR2) to rerank the documents. The second block of the table (rows 4–8) describes submissions that deployed both LR1 and LR2 as part of a seven-classifier ensemble. The seven classifiers are as follows: LR1, LR2, SVM, gradient boosting trees, stochastic gradient descent classifier, stochastic gradient descent regressor, and ridge regressor. Finally, we tried a three-classifier ensemble, with only LR2, SVM, and gradient boosting trees; this is shown in the third block of the table (rows 8–10). For the ensembles, the score from each classifier is averaged and then interpolated with the original retrieval score.

For each configuration, Table 1 also shows effectiveness in terms of standard ranked retrieval metrics. The final column denotes whether or not the run contributed to the judgment pool. We see that classifier ensembles yield better effectiveness over using only logistic regression, but it is unclear whether the seven-classifier ensemble beats the three-classifier ensemble, since the scores are all quite close. Nevertheless, given the greater complexity of a seven-classifier ensemble, the three-classifier ensemble should be preferred. In terms of the initial retrieval, axiomatic semantic term matching and RM3 yield comparable end-to-end results, although we achieve a higher AP (by a small margin) with BM25 + RM3. We do not believe any firm conclusions can be drawn about the relative merits of these query expansion techniques due to the lack of parameter tuning.

For reference, the final block of Table 1 reports results from baseline Anserini runs: BM25, BM25 with axiomatic semantic term matching, and BM25 with RM3 (respectively). We see that relevance transfer leads to large gains in effectiveness.

3 REPEATABILITY ANALYSIS

Given growing interest in reproducibility in information retrieval [3, 6, 7, 9], here we document a case study that highlights some of the challenges researchers face today. To provide a common vocabulary, we adopt the definitions of the terms *repeatable*, *replicable*, and *reproducible* as follows, per recent ACM guidelines:[3]

- Repeatable: a researcher can reliably repeat her own computation.
- Replicable: an independent group can obtain the same result using the author's own artifacts.
- Reproducible: an independent group can obtain the same result using artifacts which they develop completely independently.

Although one might imagine repeatability to be achievable in a fairly straightforward manner, in practice it is non-trivial and involves quite a number of complexities and nuances (nevermind replicability or reproducibility). The fundamental problem is that computational artifacts are *always* evolving, and even if they remains static, their execution environments can change in unanticipated ways. We detail our experiences below:

The deadline of the TREC 2018 Common Core Track was in August 2018, and our submitted runs were generated before then. Since Anserini is an open-source project, all code changes are publicly documented; however, code for relevance transfer was kept in a separate private repository. While it would have been possible to snapshot the code that generated our official submitted runs (e.g., by commit ids), unfortunately we did not do this.

The relevance transfer code was not contributed to the Anserini code repository until December 2018. However, the Anserini codebase itself had evolved from the summer, such that the contributed code targeted the most recent state of the codebase at the time (as opposed to the state of the code in August). At commit `acf4c87` (dated 12/15/2018), when we were ready to repeat the previously submitted TREC runs, we obtained different results. This point is worth emphasizing:

> *We, as the authors, were unable to repeat the results of our own submitted runs!*

In other words, the exact state of the computational artifact that generated our official TREC runs had been lost forever and cannot

[3]https://www.acm.org/publications/policies/artifact-review-badging

Run	AP (official)	AP (12/15)	AP (12/18)
`h2oloo_LR2_rm3`	0.3273	0.3539	0.3569
`h2oloo_enrm30.6`	0.3382	0.3652	0.3670

Table 2: Results of our repeatability analysis, comparing our official submissions with code at two other points in time.

be recreated. We did more rigorously document the relevance transfer code that was actually contributed to the Anserini repository,[4] and in Table 2 we provide two points of comparison. Our attempts to recreate `h2oloo_enrm30.6`, our most effective submitted run, yielded AP shown in the third column, under 12/15. For reference, we also provide AP scores for the comparable baseline with only logistic regression, `h2oloo_LR2_rm3`. We see that, for reasons that we were not able to identify, the effectiveness improved—likely as the result of general improvements to the codebase.

Shortly after incorporating the relevance transfer code, we upgraded the underlying Lucene dependency of Anserini from version 6.3 to version 7.6 (commit `e71df7a`, 12/18). This changed the effectiveness of our runs again, which is shown in the rightmost column in Table 2. Effectiveness increased again (slightly).

Our story has a happy ending because in each case, effectiveness improved. However, imagine the alternative where the effectiveness *decreased*. Would it have been "legitimate" (for example, from an ethical perspective) to report a result that is no longer achievable even though it had been gotten under some unknown conditions? By definition, an unrepeatable result is unscientific. We shudder to ponder how many results "enshrined" in the literature are not repeatable, but have gone unnoticed.

Although to some extent we are at fault for this state of affairs: for example, better record keeping could have allowed us to recover *exactly* the code that was used to submit the runs. However, even meticulous documentation efforts might not have been sufficient. For example, changes to underlying libraries such as `scikit-learn` might cause our runs to be non-repeatable. While it is possible to document and capture underlying libraries, where do we stop? In the context of neural question answering, Crane [4] recently documented a litany of details that complicate repeatability, down to compliance issues of math libraries with the IEEE 754 floating point specification. Increasingly low-level capture techniques (e.g., virtual environments, Docker, virtual machines, etc.) can address more and more of these issues, but at greater costs, slowing down progress. As an example, there are known hardware differences that affect floating point computations,[5] and hence might introduce minor perturbations in the ranked lists that in turn yield small differences in scores. How do we deal with such issues? There is a tension between repeatability (no doubt desirable) and the pace of progress. The optimal balance is difficult to find.

The fundamental challenge is that computational artifacts and execution environments are never static. It's not merely a matter of "document everything", because "everything" involves an unreasonable number of variables. Of course, we only want to document the variables that have a direct bearing on the experiment at hand, but often we don't actually know what they are—since that's the point of the inquiry to begin with. Furthermore, variables can change between experimental trials unbeknownst to researchers (for example, a system-wide upgrade of a core library by an administrator). Repeatability, as it turns out, isn't trivial.

4 CONCLUSIONS

This report documents our experiences reproducing the cross-collection relevance transfer technique proposed by Grossman and Cormack [8] and then applying it in TREC 2018. Along the way, we identified repeatability challenges, highlighting issues that likely affect other researchers in the computational sciences. We hope that our experiences contribute lessons on "how to do good science" (both positive and negative).

5 ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the 13th Text REtrieval Conference (TREC 2004).*

[2] James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel, and Ellen Voorhees. 2017. TREC 2017 Common Core Track Overview. In *Proceedings of the 26th Text REtrieval Conference (TREC 2017).*

[3] Jaime Arguello, Matt Crane, Fernando Diaz, Jimmy Lin, and Andrew Trotman. 2015. Report on the SIGIR 2015 Workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR). *SIGIR Forum* 49, 2 (2015), 107–116.

[4] Matt Crane. 2018. Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics* 6 (2018), 241–252.

[5] Hui Fang and ChengXiang Zhai. 2006. Semantic Term Matching in Axiomatic Approaches to Information Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006).* Seattle, Washington, 115–122.

[6] Nicola Ferro, Norbert Fuhr, Kalervo Järvelin, Noriko Kando, Matthias Lippold, and Justin Zobel. 2016. Increasing Reproducibility in IR: Findings from the Dagstuhl Seminar on "Reproducibility of Data-Oriented Experiments in e-Science". *SIGIR Forum* 50, 1 (2016), 68–82.

[7] Nicola Ferro and Diane Kelly. 2018. SIGIR Initiative to Implement ACM Artifact Review and Badging. *SIGIR Forum* 52, 1 (2018), 4–10.

[8] Maura R. Grossman and Gordon V. Cormack. 2017. MRG_UWaterloo and WaterlooCormack Participation in the TREC 2017 Common Core Track. In *Proceedings of the 26th Text REtrieval Conference (TREC 2017).*

[9] 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 *Proceedings of the 38th European Conference on Information Retrieval (ECIR 2016).* Padua, Italy, 408–420.

[10] Peilin Yang and Hui Fang. 2013. Evaluating the Effectiveness of Axiomatic Approaches in Web Track. In *Proceedings of the 22nd Text REtrieval Conference (TREC 2013).*

[11] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017).* Tokyo, Japan, 1253–1256.

[12] 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.

[13] Peilin Yang and Jimmy Lin. 2018. Anserini at TREC 2018: CENTRE, Common Core, and News Tracks. In *Proceedings of the 27th Text REtrieval Conference (TREC 2018).*

[14] Ruifan Yu, Yuhao Xie, and Jimmy Lin. 2019. Simple Techniques for Cross-Collection Relevance Feedback. In *Proceedings of the 41th European Conference on Information Retrieval (ECIR 2019).* Cologne, Germany.

[4]https://github.com/castorini/Anserini/blob/master/docs/runbook-trec2018-h2oloo.md

[5]https://stackoverflow.com/questions/11832428/windows-intel-and-ios-arm-differences-in-floating-point-calculations