Simple Techniques for Cross-Collection Relevance Feedback

Ruifan Yu, Yuhao Xie, and Jimmy Lin(✉)

David R. Cheriton School of Computer Science,
University of Waterloo, Ontario, Canada
jimmylin@uwaterloo.ca

Abstract. We tackle the problem of transferring relevance judgments across document collections for specific information needs by reproducing and generalizing the work of Grossman and Cormack from the TREC 2017 Common Core Track. Their approach involves training relevance classifiers using human judgments on one or more existing (source) document collections and then applying those classifiers to a new (target) document collection. Evaluation results show that their approach, based on logistic regression using word-level *tf-idf* features, is both simple and effective, with average precision scores close to human-in-the-loop runs. The original approach required inference on every document in the target collection, which we reformulated into a more efficient reranking architecture using widely-available open-source tools. Our efforts to reproduce the TREC results were successful, and additional experiments demonstrate that relevance judgments can be effectively transferred across collections in different combinations. We affirm that this approach to cross-collection relevance feedback is simple, robust, and effective.

Keywords: Relevance classifier · Logistic regression · Query expansion

1 Introduction

High-quality test collections form vital resources for guiding research in information retrieval, but they are expensive and time consuming to construct. Thus, when faced with new collections, tasks, or information needs, researchers aim to exploit existing test collections as much as possible. Learning a ranking function from one collection and applying it to another is perhaps the most obvious example, but in this paper we tackle a different use case: the transfer of relevance judgments across document collections for the *same* information need. We characterize this process as *cross-collection* relevance feedback. Suppose a user has already searched a particular document collection and the system has recorded the user's relevance judgments: Can the system then automatically take advantage of these judgments to provide a better ranking for the *same* information need on a *different* document collection? The answer is *yes*, and in this paper we reproduce and then generalize a simple yet highly effective solution using existing open-source tools.

L. Azzopardi et al. (Eds.): ECIR 2019, LNCS 11437, pp. 397–409, 2019.
https://doi.org/10.1007/978-3-030-15712-8_26

The cross-collection relevance feedback scenario can arise in a number of ways, the most common of which is when the user searches different verticals. For example, suppose the user first searches web documents and after some time, realizes that scholarly publications might better address her needs. Another case might be different sub-collections, for example, exploiting judgments on the New York Times to search the Washington Post. Yet another might be temporal segments of the same collection—for example, in a meta-analysis, a researcher might repeat the same search periodically to examine updated documents. In this paper, we focus on the case of transferring relevance judgments between sub-collections of the same genre (newswire documents), thus avoiding issues related to stylistics and genre mismatch.

Resources for studying cross-collection relevance feedback exist because various evaluation campaigns have reused topics (i.e., information needs) across different document collections at different points in time. For example, topics from the TREC 2004 Robust Track [7] were reused for the same track in TREC 2005 [8], which used a different document collection. Another more recent example is the TREC 2017 Common Core Track [2], a renewed effort to focus on the classic *ad hoc* retrieval task, which also reused topics from the TREC 2004 Robust Track. The work of Grossman and Cormack [4] achieved the highest effectiveness of all non-manual runs in the TREC 2017 Common Core Track.

The contribution of this paper is the successful reproduction of the work of Grossman and Cormack (hereafter, GC for short) for cross-collection relevance feedback. We confirm, via a reimplementation from scratch, that the simple technique proposed by GC is highly effective. Our efforts extend beyond replicability (per ACM definitions[1]), as our technical infrastructure resulted in an implementation that differed from GC in several ways. Additionally, we leverage popular open-source data science tools to provide a solid foundation for follow-on work. Finally, we generalize GC by examining different combinations of source and target collections, demonstrating that cross-collection relevance feedback reliably yields large increases in effectiveness.

2 Approach

We begin with a discussion of why GC is worth reproducing. First, the technique is extremely effective. Figure 1, reproduced from the TREC 2017 Common Core Track overview paper [2] shows the effectiveness of runs (in terms of average precision) that contributed to the judgment pools. GC is indicated by the run `WCrobust0405`, ranking third out of all submitted runs. The color coding of the figure indicates the run type: green dots (`Auto-Fdbk`) represent runs that take advantage of automatic feedback from existing judgments and blue dots (`Manual-NoFdbk`) represent manual human-in-the-loop runs. The two runs that were more effective than `WCrobust0405` involved humans who interactively searched the target collection to find relevant documents. The surprising

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

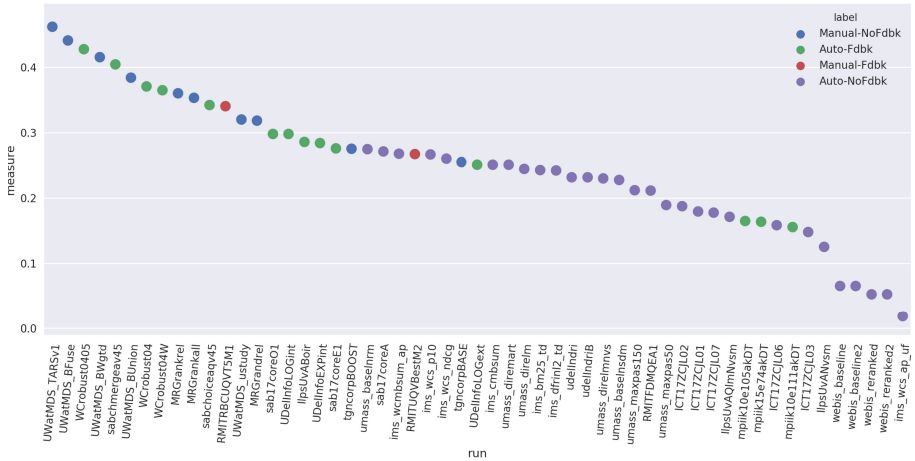

Fig. 1. Effectiveness (average precision) of runs that contributed to the pool in the TREC 2017 Common Core Track, reproduced from [2].

observation here is that relevance transfer (i.e., automatically exploiting existing judgments) approaches the effectiveness of humans manually searching the target collection. Second, the technique described by GC is very simple: the description in their overview paper is only a paragraph. This combination of effectiveness and simplicity makes GC worthy of detailed study.

At a high-level, GC trained logistic regression classifiers on the union of relevance judgments from the TREC 2004 and 2005 Robust Tracks. A separate classifier was trained for each topic, capturing notions of relevance for that specific information need. Documents were represented in terms of word-level *tf-idf* features on the union of the collections used in the 2004 and 2005 evaluations, as well as the collection used in Common Core 2017. Each logistic regression model was learned using Sofia-ML[2] and then applied to the entire Common Core collection. The top 10,000 documents, in decreasing order of classifier score, comprised the final ranked list for each topic.

To aid in our efforts, Gordon Cormack kindly supplied us with the source code used to generate the runs. However, the source code comprised a series of complex bash scripts that were not documented; although we were able to examine the code to recover the gist of its functionality, we were not able to successfully run the code to replicate the results. During our reimplementation, we did not encounter any need to specifically ask the authors questions. However, there was one important detail critical to effectiveness that was left out of their description—we were able to glean this only by looking at the source code (more details in Sect. 3).

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

Given the simplicity of the technique, instead of attempting to *exactly* reproduce GC from scratch, we made a few different design choices, discussed below:

Reranking Search Results. Instead of applying the relevance classifiers over the *entire* collection, we adopted a reranking approach where each model was applied to only the top $k = 10,000$ hits from an initial retrieval run.

Incorporating Document Scores. In the final GC submission, documents were simply sorted by classifier scores. In our case, since we were reranking documents from an initial retrieval, it made sense to combine classifier scores with the original document scores (which we accomplished via linear interpolation).

Leveraging Widely-Used Open-Source Tools. We aimed to build an implementation to serve as the foundation of future efforts, and thus decided to leverage widely-used open-source tools: the Python machine learning package `scikit-learn` and the Anserini IR toolkit [9,10]. In particular, our use of Python meant that we could take advantage of Jupyter notebooks and other modern data science best practices for interactive data exploration and manipulation.

3 Implementation

To be precise, our initial efforts focused on reproducing the run `WCrobust0405` submitted by GC for the TREC 2017 Common Core Track (henceforth, Core17 for convenience). The run leveraged relevance judgments from the TREC 2004 and 2005 Robust Tracks (henceforth, Robust04 and Robust05, respectively). Core17 used the New York Times Annotated Corpus; Robust04 used TREC Disks 4 & 5 (minus Congressional Records) and Robust05 used the AQUAINT document collection. All 50 topics in Core17 are contained in Robust04, while Core17 and Robust05 only share 33 common topics. The run `WCrobust0405` used training data from Robust04 and Robust05 (where available); relevance judgments from Core17 served as a held-out test set.

All source code for replicating results reported in this paper is available in the Anserini code repository[3] (post v0.3.0 release, based on Lucene 7.6) at commit `9548cd6` (dated Jan. 19, 2019).

The per-topic breakdown of relevance judgments is shown in Fig. 2, which plots both the volume of judged documents as well as the proportion of relevant documents. It is immediately clear that both the volume and the proportion of relevant labels vary across topics as well as collections. Furthermore, there are usually many more non-relevant judgments than relevant judgments (and this skew is especially severe for some topics). Although this observation isn't surprising, it reminds us that we are dealing with an unbalanced classification problem, and that the prior probability of relevance varies greatly.

[3] http://anserini.io/.

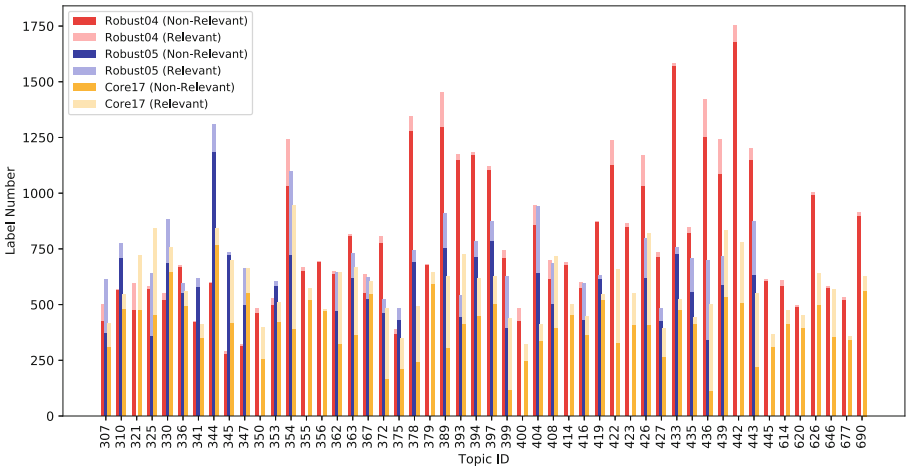

Fig. 2. Per-topic analysis of judgments from Robust04, Robust05, and Core17.

3.1 Feature Extraction and Classifier Training

We began by indexing all three collections (Robust04, Robust05, and Core17) using the Anserini IR toolkit [9,10], which is based on the popular open-source Lucene search toolkit. Anserini provides convenient tools to dump out raw *tf-idf* document vectors for arbitrary documents. Data preparation consisted of extracting these document vectors for all judgments in Robust04 and Robust05 for each topic. The features for these document vectors are comprised of stemmed terms as processed by standard Lucene analyzers. Although we extracted document vectors for each collection individually, the output is post-processed so that the final feature space is the union of vocabulary terms from the training corpora (Robust04 and Robust05 in this case). This meant that out-of-vocabulary terms may be observed at inference time on Core17 data.

Our implementation differs from GC: their brief description (in the track overview paper) suggests that their *tf-idf* document vectors are computed with respect to the union of the three collections (although this point is not explicit). In our case, eliminating *a priori* knowledge about the target collection makes our implementation more general. Furthermore, our approach leverages existing IR tools to extract document vectors, making it easier to vary source/target collections (see additional experiments later). However, we do not believe that this detail has a substantive impact on effectiveness.

As the last step, all feature vectors were converted to unit vectors by L_2 normalization. This was an important detail not mentioned in the GC description, but has a large impact on effectiveness since document lengths vary across collections. Our initial efforts did not include this normalization, and we were not able to reproduce effectiveness values anywhere close to those reported by GC. We realized this omission only after consulting the source code of the original

implementation. Perhaps in retrospect, the need for normalization is obvious, but this detail provides an example of the difficulty of reproducibility, where small implementation decisions make a big difference.

The feature vectors prepared in the manner described above were then fed to the Python machine learning package `scikit-learn` [6]. Each topic was treated as an independent training dataset to learn a relevance classifier for that particular information need. One advantage of using `scikit-learn` is that we can easily explore a wide range of different models. We did in fact do so, but discovered that different models as well as variations within families of models (for example, different loss functions, regularization methods, and optimization algorithms) did not make much of a difference in terms of effectiveness. For brevity, we decided to report results with three representative models:

- Logistic regression (LR). We used the so-called "balanced" mode to automatically adjust class weights to be inversely proportional to class frequencies.
- Support vector machines (SVM). We used a linear kernel and the "balanced" mode as well.
- Gradient-boosted decision trees (GB Tree). Specifically, LightGBM [5].

In addition to evaluating each model individually, we also explored an ensemble of all models using simple score averaging.

3.2 Reranking Retrieval Results

Classifiers trained in the manner described above capture relevance with respect to an information need at the lexical level, which can then be applied to a new (target) collection to infer document relevance with respect to the same information need. GC accomplished this by applying inference on *every* document in the target collection and generating a ranked list based on the classifier scores. While this approach is feasible for newswire collections that are moderate in size, especially with an efficient classifier implementation, scaling to larger collections is potentially problematic. Classifying every document is also computationally wasteful, since most of the documents in a collection will not be relevant.

Instead, we adapted GC into a reranking architecture, where the relevance classifier is used to rescore an initial candidate list of documents generated by traditional *ad hoc* retrieval techniques. In our case, we used title queries from the topics to produce the top $k = 10,000$ results using two query expansion techniques: RM3 [1] and axiomatic semantic term matching [3] (Ax for short). In both cases, we used default parameters in the Anserini implementation. Query expansion techniques provide the classifier with a richer set of documents to work on, thus potentially enhancing recall.

We applied our relevance classifiers to these initial results to generate a final ranking in two different ways: First, by ignoring the RM3 and Ax retrieval scores and reranking solely on the classifier scores. Second, by a linear interpolation between retrieval and classifier scores as follows:

$$\text{score} = \alpha \cdot \text{score}_{\text{classifier}} + (1 - \alpha) \cdot \text{score}_{\text{retrieval}}$$

Table 1. Baseline retrieval results.

Method	Robust04		Robust05		Core17	
	AP	P10	AP	P10	AP	P10
BM25	0.1442	0.3280	0.2046	0.4818	0.1977	0.4920
BM25+RM3	0.1725	0.3500	0.2716	0.5333	0.2682	0.5560
BM25+Ax	0.1779	0.3560	0.2699	0.5121	0.2700	0.5680

The first case can be viewed as a special case of the second where $\alpha = 1$. The interpolation parameter can be learned by cross validation, but experiments show that results are not particularly sensitive to the setting.

Beyond our attempt to reproduce the `WCrobust0405` run, we also ran experiments that considered different combinations of source and target document collections to examine the generality and robustness of GC.

4 Experimental Results

We first establish baselines on Robust04, Robust05, and Core17. Effectiveness measured in terms of average precision (AP) at rank 1000 and precision at rank 10 (P10) is shown in Table 1 for title queries (in all our experiments we ignored the descriptions and narratives). The rows show effectiveness with "bag of words" BM25, BM25 combined with RM3 expansion [1], and BM25 with axiomatic semantic term matching [3]; all used default Anserini parameters. Note that for Robust04 and Core17, metrics are computed over the 50 common topics, while for Robust05, metrics are computed over the 33 common topics. Consistent with the literature, query expansion yields sizeable gains in effectiveness. We find that RM3 is slightly more effective than axiomatic semantic term matching.

Table 2 shows results from our relevance transfer experiments to reproduce `WCrobust0405`: training on Robust04 and Robust05 judgments, evaluating on Core17 judgments. The table reports results applied to the initial ranked list from RM3 (left) and axiomatic semantic term matching (right); baseline effectiveness is reported in the second row (copied from Table 1). The effectiveness of `WCrobust0405` is presented in the first row. The remaining parts of the table are organized into three blocks: The first presents results where we ignore the retrieval scores and sort by the relevance classifier scores only. The second shows results from interpolating the original retrieval scores and the classifier scores, with the optimal interpolation weight α (i.e., provided by an oracle, in tenth increments, selected separately for each metric). The third block shows interpolation results with a weight of $\alpha = 0.6$. Within each block, individual rows show the effectiveness of each model; we also show results of the ensemble using simple score averaging (denoted "All Classifiers").

Focusing on optimal α values (we examine sensitivity to the interpolation weight below), we see that our results successfully reproduce the technique of GC: We achieve comparable effectiveness and demonstrate large increases over

Table 2. Relevance transfer results: train on Robust04 and Robust05, test on Core17.

	RM3		Axiomatic	
	AP	P10	AP	P10
`WCrobust0405`	0.4278	0.7500	0.4278	0.7500
Baseline	0.2682	0.5560	0.2700	0.5680
Classification Only				
LR	0.3721	0.7420	0.3605	0.7440
SVM	0.3595	0.7440	0.3445	0.7340
GB Tree	0.3069	0.6640	0.3046	0.6660
All Classifiers	0.4011	0.7700	0.3907	0.7660
Interpolation (Optimal α)				
LR	0.4198	0.7720	0.4166	0.7840
SVM	0.4153	0.7640	0.4135	0.7780
GB Tree	0.3857	0.7320	0.3945	0.7460
All Classifiers	0.4452	0.7780	0.4472	0.7840
Interpolation ($\alpha = 0.6$)				
LR	0.4198	0.7640	0.4166	0.7700
SVM	0.4153	0.7580	0.4121	0.7740
GB Tree	0.3815	0.7320	0.3893	0.7460
All Classifiers	0.4451	0.7540	0.4472	0.7740

the baselines; the absolute values of the metrics are quite close. Based on a paired t-test (which we use throughout this paper for testing statistical significance, at the $p < 0.01$ level), we find no significant differences between any of our models and `WCrobust0405` in terms of both AP and P10.

Interestingly, a classification-only approach does not appear to be effective in our reranking implementation. For both RM3 and axiomatic semantic term matching, weighted interpolation with optimal α is significantly better than the classification-only approach in terms of average precision (across all models and the ensemble), but not significantly better in terms of P10 (except for GB Tree).

In terms of different models, we observe that logistic regression (LR) and SVM yield comparable results. None of the differences (for both metrics, for both initial rankings) are statistically significant. The tree-based model (GB Tree) performs quite a bit worse than either LR or SVM; these differences, however, are not significant with the exception of GB Tree vs. SVM in terms of AP. Finally, an ensemble using simple score averaging yields effectiveness that is higher than any individual model; these differences are significant for AP, but not P10. Comparing RM3 vs. axiomatic semantic term matching, we find that differences in both AP and P10 are not significant.

An optimal interpolation weight α assumes the existence of an oracle, which is of course unrealistic in a real-world setting. To address this issue, we performed

a sensitivity analysis by varying α from zero to one in tenth increments, with the results shown in Fig. 3. As expected, the curve has a convex shape, with a peak in a fairly wide range, from 0.5 to 0.7. We further ran a five-fold cross-validation experiment: training on Robust04 and Robust05 as before, but selecting a fifth of the test topics from Core17 as a validation set to select α and evaluating on the remaining topics. In each case, the optimal weight lies in this 0.5 to 0.7 range (although the exact value varies from fold to fold). From this cross-validation analysis, we conclude that 0.6 appears to be a reasonable interpolation weight that can be adopted in the absence of validation data. In Table 2, the third block of rows report results with $\alpha = 0.6$, and we see that effectiveness is quite close to the optimal settings. None of the differences in effectiveness between optimal α and $\alpha = 0.6$ are statistically significant. We further demonstrate the robustness of this setting in experiments below.

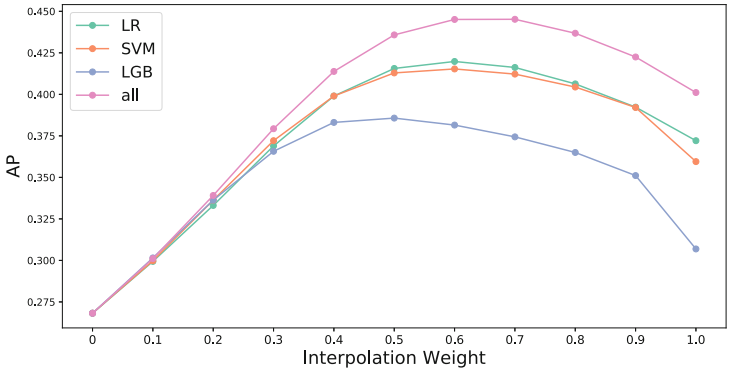

Fig. 3. AP scores with different interpolation weights.

The work of GC represents one specific instance of cross-collection relevance feedback: leveraging judgments from Robust04 and Robust05 to improve ranking effectiveness on Core17. Of course, given the available evaluation resources, it is possible to examine different combinations of source and target document collections. Such experiments allow us to examine the generality of the technique: results are reported in Table 3. Here, we treat BM25+RM3 as the baseline and the initial ranking. For simplicity, we fixed the relevance classifier to logistic regression interpolated with BM25+RM3 ($\alpha = 0.6$), denoted BM25+RM3+LR. The first column denotes the target collection used for evaluation and the second column denotes the source of the relevance judgments used for training. The first three rows are simply repeated from Table 2 for convenience. In addition to training on both Robust04 and Robust05 data together, we also tried each collection separately. Training on Robust04 alone actually corresponds to the run `WCrobust04` submitted by GC, whose effectiveness we repeat here for convenience. Note that when testing on Core17 and Robust04, the evaluation is conducted over 50 topics in all cases, and on Robust05, over 33 topics.

Table 3. Results on different combinations of source and target collections.

Test	Train	Approach	AP	P10
Core17	-	BM25+RM3	0.2682	0.5560
Core17	Robust04, Robust05	`WCrobust0405`	0.4278	0.7500
Core17	Robust04, Robust05	BM25+RM3+LR	0.4198	0.7640
Core17	Robust04	`WCrobust04`	0.3711	0.6460
Core17	Robust04	BM25+RM3+LR	0.3812	0.7360
Core17	Robust05	BM25+RM3+LR	0.3721	0.7060
Robust04	-	BM25+RM3	0.1725	0.3500
Robust04	Robust05, Core17	BM25+RM3+LR	0.3520	0.6060
Robust04	Robust05	BM25+RM3+LR	0.2802	0.5040
Robust04	Core17	BM25+RM3+LR	0.3248	0.5700
Robust05	-	BM25+RM3	0.2716	0.5333
Robust05	Robust04, Core17	BM25+RM3+LR	0.4471	0.7515
Robust05	Robust04	BM25+RM3+LR	0.3647	0.6970
Robust05	Core17	BM25+RM3+LR	0.4042	0.7242

These results generalize the classification-based relevance transfer technique of GC by demonstrating consistent and large effectiveness gains with different source and target collections. We find that the technique is both simple and robust. Moreover, results show that more relevance judgments yield higher effectiveness, even if those judgments come from different collections: training on two source collections consistently beats training on a single collection. Note that in these experiments, we used a single interpolation weight ($\alpha = 0.6$) and performed no parameter tuning. This further validates the recommendation derived from the results in Table 2.

Our final set of experiments consists of in-depth error analyses to better understand the impact of relevance transfer. Figure 4 presents per-topic analyses, comparing the effectiveness (in terms of average precision) of logistic regression interpolated with BM25+RM3 ($\alpha = 0.6$) with the BM25+RM3 baseline. Each bar represents a topic and the bars are sorted by differences in AP. We show evaluation on Core17 in the top plot, Robust04 in the middle plot, and Robust05 in the bottom plot (using all available judgments).

As is common with many retrieval techniques, relevance transfer improves many topics (some leading to spectacular improvements) but hurts some topics as well. Our implementation only decreased effectiveness for two topics on Robust05, but that test set contains fewer topics overall, so we hesitate to draw any definitive conclusions from this. Focusing on the top plot (train on Robust04 and Robust05, test on Core17), we performed manual error analysis to try and understand what went wrong. Topic 423, the rightmost bar and the worst-performing topic, is simply the named entity "Milosevic, Mirjana Markovic".

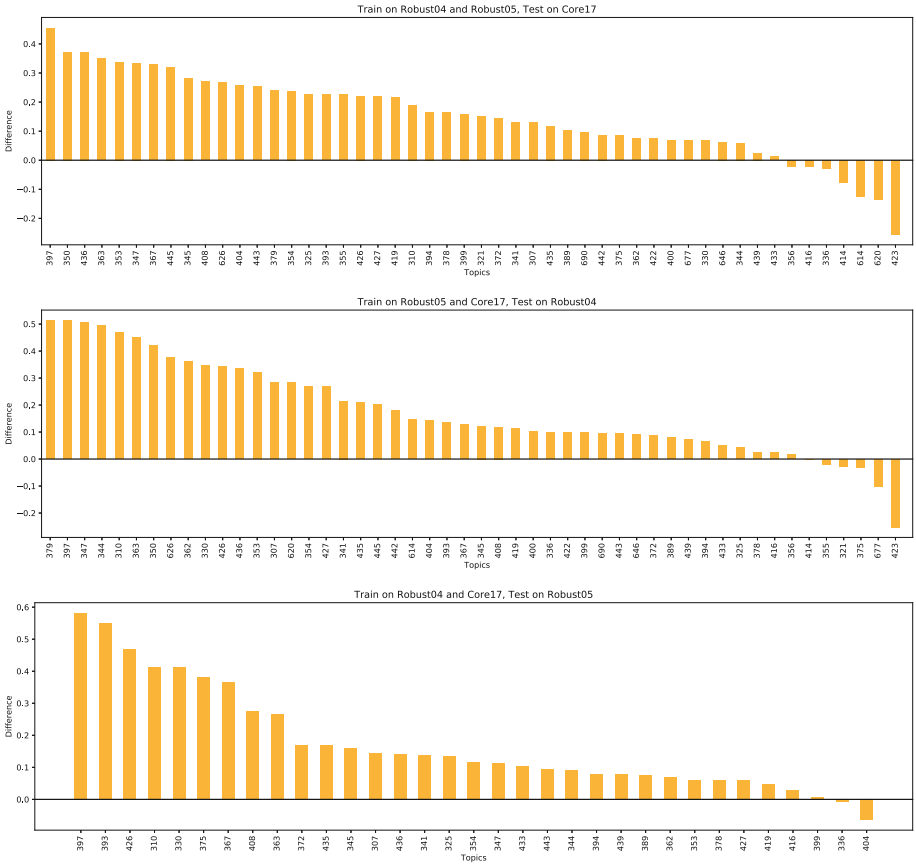

Fig. 4. Per-topic analysis, comparing interpolated logistic regression with BM25+RM3 and the BM25+RM3 baseline.

For this topic, BM25+RM3 achieves AP 0.8252; interpolated relevance classification yields AP 0.5698. Topic 620, the second worst-performing topic, is "France nuclear testing": BM25+RM3 achieves AP 0.7716, while relevance classification drops AP down to 0.6358. For this classifier, the highest-weighted feature is the term "Greenpeace", the non-profit environmental organization. This term leads the classifier astray likely because of the different time spans of the collections and specific occurrences of events. In the Robust04 and Robust05 collections, French nuclear testing was frequently associated with Greenpeace protests; in Core17, this association does not appear to be as strong. We might characterize this as an instance of relevance drift, where notions of relevance shift over time and across collections.

We further observe that both topics are relatively "easy", given the high average precision scores of the baselines. This suggests that relevance transfer has the potential to "screw up" easy topics, which is not a unique

characteristic of this technique. In general, query expansion runs the risk of decreasing effectiveness on topics with already high scores, since scores can only further increase by bringing in additional relevant documents. Any bad expansion term can depress the rankings of relevant documents, thereby decreasing the overall score.

Looking at all three plots with different target collections, it is difficult to draw any firm conclusions. Comparing Core17 (top) and Robust04 (middle), we see that topic 423 performs poorly in both cases. Unfortunately, that topic is not in the overlap set with Robust05, so the result is missing from the bottom plot. However, it is *not* the case that topics perform poorly in a consistent manner—when evaluating on Core17, topic 620 is the second worst-performing topic, but when evaluating on Robust04, we observe a large effectiveness improvement. The choice of source and target collections appears to have a large impact on effectiveness differences in relevance transfer.

5 Conclusion

As a succinct summary of our results, we find that the cross-collection relevance feedback technique of GC "works as advertised". Additional experiments further demonstrate its generality across different combinations of source and target collections. We conclude that this technique is simple, robust, and effective. In addition to these experimental findings, the concrete product of our effort is an open-source computational artifact for replicating our experiments, implemented with modern tools (Lucene and `scikit-learn`) that can serve as the foundation for future work.

Although our experiments demonstrate the generality of relevance transfer with simple "bag of words" classifiers, we believe that more work is needed to better understand when an information need can benefit from existing relevance judgments on another collection. At the core, the problem formulation is one of document classification. Thus, an obvious next step is to apply the plethora of techniques involving deep learning and continuous word representations to tackle this problem. We have already begun explorations along these lines.

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

References

1. Abdul-Jaleel, N., et al.: UMass at TREC 2004: novelty and HARD. In: Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004). Gaithersburg, Maryland (2004)
2. Allan, J., Harman, D., Kanoulas, E., Li, D., Gysel, C.V., Voorhees, E.: TREC 2017 common core track overview. In: Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017). Gaithersburg, Maryland (2017)

3. Fang, H., Zhai, C.: 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, pp. 115–122. ACM, New York (2006). http://doi.acm.org/10.1145/1148170.1148193
4. Grossman, M.R., Cormack, G.V.: MRG_UWaterloo and WaterlooCormack participation in the TREC 2017 common core track. In: Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017). Gaithersburg, Maryland (2017)
5. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, pp. 3146–3154 (2017)
6. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
7. Voorhees, E.M.: Overview of the TREC 2004 Robust Track. In: Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004). Gaithersburg, Maryland (2004)
8. Voorhees, E.M.: Overview of the TREC 2005 Robust Track. In: Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005). Gaithersburg, Maryland (2005)
9. Yang, P., Fang, H., Lin, J.: Anserini: enabling the use of Lucene for information retrieval research. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR 2017, pp. 1253–1256. ACM, New York (2017). http://doi.acm.org/10.1145/3077136.3080721
10. Yang, P., Fang, H., Lin, J.: Anserini: reproducible ranking baselines using Lucene. J. Data Inf. Qual. **10**(4), Article 16 (2018)