Different Rankers on Different Subcollections

Timothy Jones[1], Falk Scholer[1], Andrew Turpin[2], Stefano Mizzaro[3], and Mark Sanderson[1]

[1] RMIT University
[2] University of Melbourne
[3] University of Udine

Abstract. Recent work has shown that when documents in a TREC ad hoc collection are partitioned, different rankers will perform optimally on different partitions. This result suggests that choosing different highly effective rankers for each partition and merging the results, should be able to improve overall effectiveness. Analyzing results from a novel oracle merge process, we demonstrate that this is not the case: selecting the best performing ranker on each subcollection is very unlikely to outperform just using a single best ranker across the whole collection.

Keywords: Collection Partitioning, Subcollections, Retrieval effectiveness.

1 Introduction

Recent work by Sanderson et al. [7] and Jones et al. [3] showed that when TREC collections are partitioned based on the source of a document (e.g. LA Times, Financial Times, etc.) and retrieval systems are evaluated on those partitions, the ordering of the systems differs significantly across the partitions. Their results show that there are some systems that are more effective on some partitions of the TREC collections than others.

The researchers hypothesize that if a document collection can be partitioned in such a way that specific rankers work well on specific partitions, then it should be possible to merge the results from the selected rankers, producing a system that overall is more effective than a single state-of-the-art ranker retrieving from the whole collection. This hypothesis is explored in this paper.

2 Related Work

One of the first works to investigate fusing results from multiple runs to improve effectiveness was by Fox and Shaw [2]. Their work focused on combining similarity scores produced by multiple retrieval strategies. They tested six combination strategies, and showed improvement over a single run strategy. Of particular note, the researchers described the combination strategies *CombSUM* (summing the similarity scores for each document) and *CombMNZ* (multiplying the sum by the number of systems that gave the document a non-zero score), both of which were found to result in improvements over a single run.

Beitzel et al. note that while this type of data fusion is sometimes effective, it is not understood why or where it is effective [1]. Previously, it had been hypothesized

A. Hanbury et al. (Eds.): ECIR 2015, LNCS 9022, pp. 203–208, 2015.

Table 1. The sixteen Terrier rankers used in the reported experiments

A) BB2c1	B) BM25b0	C) DFR_BM25c1	D) DFRee_1
E) DLH13_9	F) DLH_8	G) DPH_0	H) Hiemstra_LM0
I) IFB2c1	J) In_expB2c1	K) In_expC2c1	L) InL2c1
M) LemurTF_IDF_12	N) LGDc1	O) PL2c1	P) TF_IDF_15

that data fusion was effective where there was a greater overlap of relevant documents than overlap of non-relevant documents between lists [5]. However, using a series of experiments where system differences (indexer, stemmer, word definition, etc) were constant, but ranking strategies varied, Beitzel et al. show that where the document lists of runs were similar, fusion was unlikely to show improvement—as scores were in effect scaled by the fusion process. Instead, fusion is more effective when new relevant documents are introduced.

More recent merging research has been conducted. LambdaMerge, was described by Sheldon et al. [8]. The technique produces a more effective merged list than simpler techniques, or any of the single strategy source lists. Wu et al. [11] examined data fusion when including evidence from anchor text in web pages. They note that data fusion can be broken into two categories: *search result fusion* (using the score or ranking of a document to produce the final ranking), and *evidence fusion* (using multiple types of evidence as input to the ranking function). They showed that the most effective fusion method from each class produced a significant improvement over baseline retrieval, but that there was no significant difference between the best fusion methods.

3 Methods and Data

To investigate the hypothesis of our paper we require: test collections, collection partitions, a set of rankers, and a rank merging strategy.

Collections: This research investigates a hypothesis about certain properties of collections that were established in past work [3, 7]. We therefore use the same ad hoc collections from TREC 4-8 here. We partition the TREC collections into subcollections based on the publication *source* of the documents (e.g. Financial Times, LA Times, Federal Register, etc.) because this was the style of partitioning used by Sanderson et al. [7]. We also examined a partitioning based on document *length*, due to the work of Wilkie and Azzopardi [10], who showed rank inconsistencies are common when document length is varied. The latter partition produces four equal-sized subcollections. Both partitions show disagreement on the best ranker, measured using the methodology of [3]. These subcollections were selected because they strongly disagreed about the best ranker–which indicates that any possible improvement due to disagreement should be large.

Rankers: As rankers, we used sixteen different parameterizations of the Terrier system [6]. Table 1 details the names of those rankers, which include variants of Language Modeling (LM); Divergence From Randomness (DFR), BM25 and TF_IDF. These are the same rankers used by Jones et al. [3].

Merging approaches: Two merging strategies are used: CombSUM [2], and a novel oracle merging scheme that preserves the order of documents between relevant documents in the lists to be merged, but optimally chooses the order of lists from which to merge. The latter scheme provides a reasonable upper bound on the effectiveness of any merging algorithm.

4 Experiments and Results

Initial approaches to improve overall effectiveness by leveraging the best performing ranker on each subcollection were not successful. Due to lack of space we do not describe this initial work here — instead, we show the results of exhaustively searching all possible combinations of rankers across the different partitioning strategies. This produces distributions of effectiveness measures for different combinations of rankers. With sixteen rankers and four partitions, for each collection splitting scheme we have 16^4 possible combinations, each merged by normalizing document scores linearly, and then ranking by the normalized scores.

We conduct an exhaustive search of ranker combinations on the TREC 8 collection, and compare effectiveness scores (MAP) with the best ranker applied across the full TREC 8 collection, which was 0.257.

Figure 1 shows the distribution of MAP scores for the combinations on the source-based partitions. The plots each show the same data, from the perspective of a different subcollection. Each plot indicates the range of scores achievable when using a particular ranker for that subcollection. The A boxplot in top right of figure 1 shows the scores of all combinations with ranker A in subcollection 1, the B all combinations with ranker B in subcollection 1, etc. In general, there is little difference between the rankers. The main result, however, is that the maximum MAP value over any combination is never higher than 0.21, substantially lower than single ranker effectiveness which is 0.257.

Figure 2 shows the results for the length-based partitioning of the collection. Although the maximum MAP is higher than for the source-based partitions, even with an exhaustive search of combinations of rankers, all MAP scores are below 0.25.

Despite evidence in past work to suggest that such partitioning might result in improvements in effectiveness, it would appear that it does not. If a particular ranker (or group of rankers) were the best for a given subcollection, one would expect it to show up as highly effective in Figures 2 or 1. While this does not appear to be the case, it can be seen that some rankers perform especially badly on particular subcollections—for example, ranker `H` in Figure 1 or ranker `O` in Figure 2.

However, these results do not conclusively show that the different ranker per subcollection strategy is failing to work, it is possible that the low scores are due to an ineffective merging strategy. This was investigated next.

4.1 An Upper Bound on Improved Retrieval

To examine an upper bound on retrieval after ranker selection has been made, we follow the best possible performance methodology introduced by Thomas and Shokouhi [9], where design choices for an element of a retrieval system are compared by assuming

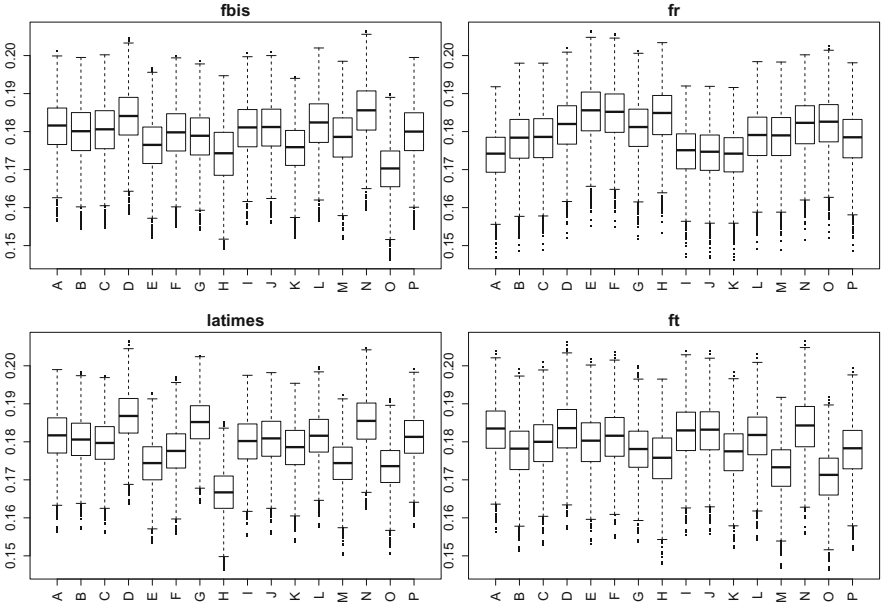

Fig. 1. MAP@1000 scores for all 16^4 ranker combinations separated according to the ranker used on each subcollection for TREC 8. Scores combined by linear normailsation. Letters A–P indicate the ranker selected for that subcollection, box plots indicate the range of scores achieved when all combinations of rankers for the other subcollections are tried.

best possible performance from all subsequent elements of the system. Here, we introduce a merging strategy called *perfect merge* (PM), which operates with the following principles (assuming rankings have no overlap). It does not violate the ordering from the ranked lists to be merged: if document a occurs below document b in one of the ranked lists, then it must occur in that order in the final list. PM does not skip any items: if document a appears at rank 1 in a ranked list, then no documents below rank 1 in that list can be selected until document a has been selected. A PM of a set of ranked lists is defined as the merge that achieves the highest score under a particular evaluation measure. In this work, we use MAP@10, since evaluating deeper in the result list produces a much larger state space that is computationally expensive to compute.

Calculating all possible merges of a set of input rankings to determine the PM would be infeasible. Fortunately, the criteria above allow us to make some assumptions that reduce the determination of a PM into a natural fit for a branch and bound solution [4]. Branch and bound is a state space search that calculates the highest possible outcome from a branch of the state tree, and ignores that branch if the outcome cannot be better than the best known solution so far.

Since we are not allowed to skip any items in the merge process, the candidate solutions must start with the document at rank one in one of the source lists. Additionally, since we know the number of relevant documents that any solution can contain, and

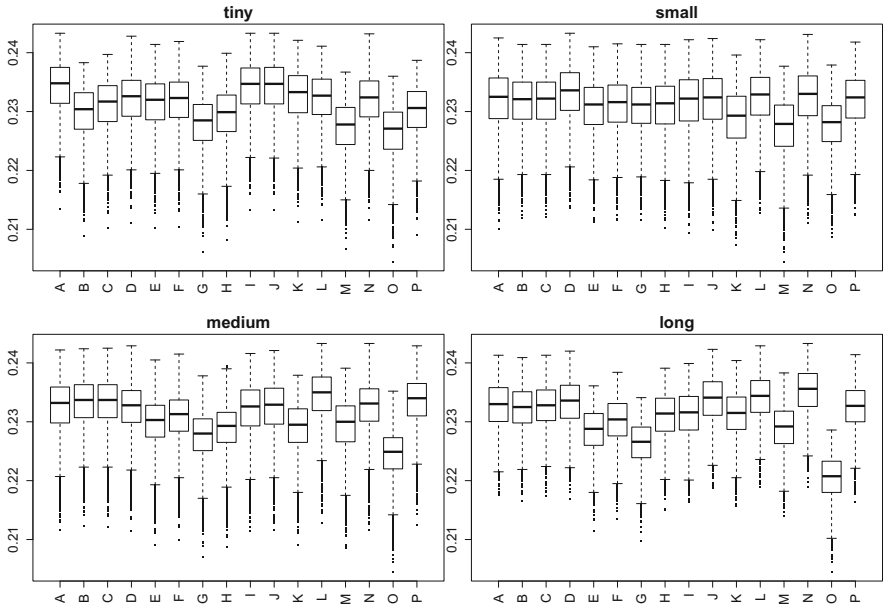

Fig. 2. MAP@1000 scores from length based subcollections displayed as Figure 1

since the best possible score for any partial ranked list can be obtained by putting all remaining relevant documents next in any partial list, we can produce an upper bound on the possible score of a partial solution. This allows us to rapidly eliminate solutions that will score poorly, and quickly converge on the optimal solution. We initialize the lower bound as the highest scoring run from any subcollection, since each input ranked list is also a candidate solution.

4.2 Ranker Selection Strategies

To simulate the best possible performance in a search system that selects different rankers for each subcollection, we assign the best performing ranker to each subcollection, measured using MAP@1000. Runs from each subcollection are then merged using PM. We call the resulting score *HybridPM*.

Since this strategy is an oracle, we need an oracle baseline to compare with. We pick one ranker that has the best mean MAP@1000 score across all subcollections. Then, we run the results for each query on each subcollection–using this one ranker–through the perfect merging process described above. We call this score *TraditionalPM*.

Table 2 shows the comparison of HybridPM and TraditionalPM on TREC 4–8 on source-based subcollections. What we find is that HybridPM does not significantly outperform TraditionalPM (paired t-test, $p > 0.05$). Under ideal merging, there is no advantage in picking different (best) rankers per subcollection.

Table 2. MAP@10 of TraditionalPM and HybridPM on source-based subcollections

	TREC 4	TREC 5	TREC 6	TREC 7	TREC 8
TraditionalPM	0.5464	0.3643	0.5205	0.4914	0.5593
HybridPM	0.5809	0.3537	0.5309	0.4926	0.5643
Percentage improvement	6.3%	2.9%	1.9%	0.2%	0.8%

5 Conclusions

Using different rankers on different subsets of a collection intuitively sounds like an approach that can improve overall search effectiveness, since previous work using the same collections and rankers had shown significant disagreement on which ranker was best. Using a simple merge, no improvements were found. Hypothesizing that this was due to the merging step, we introduced perfect merge, a strategy for producing an idealized merge of a set of input result lists. Even with perfect merge, using different rankers on each subcollection showed no improvement over using a single ranker. It can therefore be concluded that on the TREC ad hoc collections, it is very unlikely that using different rankers on each subcollection will improve retrieval. A limitation of this analysis may be that the subcollections of a TREC collection are too similar. An avenue for future work is to use subcollections that have greater variation in style and content.

References

[1] Beitzel, S.M., Jensen, E.C., Chowdhury, A., Grossman, D., Goharian, N., Frieder, O.: Recent results on fusion of effective retrieval strategies in the same information retrieval system. In: Callan, J., Crestani, F., Sanderson, M. (eds.) SIGIR 2003 Ws Distributed IR 2003. LNCS, vol. 2924, pp. 101–111. Springer, Heidelberg (2004)
[2] Fox, E.A., Shaw, J.A.: Combination of multiple searches. Proc. TREC 2, 243–252 (1994)
[3] Jones, T., Turpin, A., Mizzaro, S., Scholer, F., Sanderson, M.: Size and source matter: Understanding inconsistencies in test collection-based evaluation. In: CIKM 2014 (2014)
[4] Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica 28(3), 497–520 (1960)
[5] Lee, J.H.: Analyses of multiple evidence combination. In: Proc. SIGIR, pp. 267–276. ACM Press, New York (1997)
[6] Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A high performance and scalable information retrieval platform. In: Proc. OSIR Workshop, pp. 18–25 (2006)
[7] Sanderson, M., Turpin, A., Zhang, Y., Scholer, F.: Differences in effectiveness across subcollections. In: CIKM 2012, pp. 1965–1969. ACM (2012)
[8] Sheldon, D., Shokouhi, M., Szummer, M., Craswell, N.: Lambdamerge: Merging the results of query reformulations. In: WSDM 2011, pp. 795–804. ACM, New York (2011)
[9] Thomas, P., Shokouhi, M.: Evaluating server selection for federated search. In: Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., van Rijsbergen, K. (eds.) ECIR 2010. LNCS, vol. 5993, pp. 607–610. Springer, Heidelberg (2010)
[10] Wilkie, C., Azzopardi, L.: Efficiently estimating retrievability bias. In: de Rijke, M., Kenter, T., de Vries, A.P., Zhai, C., de Jong, F., Radinsky, K., Hofmann, K. (eds.) ECIR 2014. LNCS, vol. 8416, pp. 720–726. Springer, Heidelberg (2014)
[11] Wu, M., Hawking, D., Turpin, A., Scholer, F.: Using anchor text for homepage and topic distillation search tasks. JASIST 63, 1235–1255 (2012)