Exploiting Global Impact Ordering for Higher Throughput in Selective Search

Michał Siedlaczek[0000-0002-9168-0851], Juan Rodriguez[0000-0001-6483-6956], and Torsten Suel[0000-0002-8324-980X]

Computer Science and Engineering, New York University, New York, US
`{michal.siedlaczek, jcr365@nyu.edu, torsten.suel}@nyu.edu`

Abstract. We investigate potential benefits of exploiting a global impact ordering in a selective search architecture. We propose a generalized, ordering-aware version of the learning-to-rank-resources framework [9] along with a modified selection strategy. By allowing partial shard processing we are able to achieve a better initial trade-off between query cost and precision than the current state of the art. Thus, our solution is suitable for increasing query throughput during periods of peak load or in low-resource systems.

Keywords: Selective search, Global ordering, Shard selection

1 Introduction

Substantial advances have recently been made in selective search—a type of federated search where a collection is clustered into topical shards, and a *selection algorithm* selects for each query a small set of relevant shards for processing. The latest state of the art proposed by Dai et al. [9] uses a learning-to-rank technique from complex document ranking [18] to select shards during query processing.

We propose a generalization of this approach that exploits a global impact ordering of the documents, in addition to topic-based clustering. Query-independent global impact scores model the overall quality of documents in a collection. They have previously been used in unsafe early termination techniques, such as tiering, which limit query cost by disregarding documents deemed unimportant, unless a query is found to require more exhaustive processing. However, to our knowledge, this is the first attempt at combining selective search and global ordering-based early termination.

Contributions. We produce global orderings for GOV2 and Clueweb09-B collections; describe a modified selective search architecture that can exploit the orderings, and expand the state-of-the-art solution to allow for partial shard selection; compare our results with the baseline, and show that we achieve a better quality-efficiency trade-off at low query costs; discuss a number of research opportunities that are motivated by our work.

2 Related Work

Selective Search. A number of researchers have recently explored selective search architectures, where a collection is clustered into topical shards [10, 16]. Then, when processing a query, we select a limited number of shards that are predicted to be most relevant to the query topic. The literature identifies three classes of shard selection algorithms: Term-based methods [1, 7] model the language distribution of each shard. Sample-based methods [16, 24, 25] use results from a centralized sample index, which contains a small random sample (up to 2%) of the entire index. Supervised methods [9, 13] use labeled data to learn models that predict shard relevance. The current state of the art, proposed by Dai et al. [9], belongs to the last group. We derive our solution from their model, and also use it as a baseline for comparison.

Global Impact Ordering and Index Tiering. A global impact ordering is a query-independent ordering of the documents in a collection in terms of quality or importance. There are a number of ways to compute such orderings, including Pagerank [20], spam scores [8], performance of a document on past queries [2, 12], or machine-learned orderings [22]. Global orderings can be exploited for faster query processing by organizing index structures such that higher-quality documents appear first during index traversal.

One well known approach that exploits a global ordering is called *index tiering* [4, 17, 19, 21, 23]. Here, the collection is partitioned into two or more subsets called tiers based on document quality, and an inverted index is built for each of them. Each query is then first evaluated on the first tier with the highest-quality documents, and only evaluated on additional tiers if results on earlier tiers are considered insufficient. An alternative approach simply assigns IDs to documents in descending order of quality, and then stops index traversal once enough high quality documents have been evaluated for a query [3, 12]. We refer to this method as *global rank cutoff* (GRC).

Our Approach. We apply GRC to a selective search environment by ordering documents inside each cluster by global impact ordering. Our ordering is determined by the performance of documents on past queries as in [2, 12], which appears to provide a stronger ordering than Pagerank or spam scores, though limited improvements may be possible with an ML-based approach that combines a number of features [22]. We note that index tiering and GRC are orthogonal and complementary to safe early termination techniques such as WAND [6], BMW [11], or Max-Score [26]. In fact, previous work [15] has shown that these techniques provide benefits in selective search architectures, and we would expect them to also be profitable in our proposed architecture.

3 Ordering-Aware Selective Search

Clustering. Our index relies on the collection of documents being clustered into a number of topical shards. Since this step is orthogonal to our work, we adopt the clusters used by Dai et al. [9, 10].

Reordering. Documents in each shard are reordered according to a global ordering obtained by counting document hits on a query log: Given a set of queries Q and a document d, we define a hit count $h(d) = \sum_{q \in Q} I_q(d)$, where $I_q(d)$ is equal to 1 if d is among the top k results retrieved for query q, or 0 otherwise (we used $k = 1000$). For Clueweb09-B, we used 40,000 queries from the TREC 2009 Million Query Track. For GOV2, we used a set of 2 million queries randomly generated by a language model learned from a sample of several TREC query tracks. In each shard, we then assign document IDs to documents based on decreasing hit count. Evaluation is performed on a set of queries that is disjoint from those used to compute the ordering or train the language model.

Ordered Partitioning. The idea behind reordering is to facilitate global rank cutoff within topical clusters. In this paper, we approximate it by partitioning consecutive document IDs in each shard into b ranges of equal size, effectively setting up b cutoff points where processing can stop. We call these partitions *buckets* and number them with consecutive integers, where 1 denotes the first (highest quality) bucket and b denotes the last one. This approach allows us to extend selective search with global ordering while trading off model accuracy and complexity. Large values of b model GRC well but make both learning and feature extraction more expensive. Structures with small b are fast and resemble discrete tiers. When $b = 1$, we get the standard selective search approach, as done in previous work. From this point forward, we refer to a solution with b buckets as B_b. In particular, B_1 denotes the baseline.

Learning Model. We modify the learning-to-rank-resources model proposed by Dai et al. [9]; see that paper for more details. We discard CSI-based features, as they were shown to provide only negligible improvement but are expensive to compute [9]. To the shard-level features (shard popularity and term-based statistics), we add only one bucket-level feature, namely the bucket number, from 1 to b. Thus, we train and predict a ranking of buckets instead of shards. This ranking is further converted into the final selection as described below.

Bucket Cost. Given a query, every bucket i in a shard s has an associated processing cost $c(s, i)$. We consider two *cost models*: (1) uniform cost: $c(s, i) = 1/b$, used during shard selection, and (2) posting cost: the number of postings for the query terms within bucket (s, i), used during evaluation. We also present one set of experiments where we add an additional per-shard access cost to the posting cost, in order to model the overhead of forwarding a query to a shard.

Shard Selection. In constrast to previous shard selection algorithms, our approach requires us to select shards as well as a cut-off point for early termination inside each selected shard (Figure 1). Our model predicts a bucket-level ranking, but to model GRC, we need to make sure to select a prefix of the bucket list in each shard. Thus, given a budget T per query, we iterate over all buckets from highest to lowest ranked, until the remaining query budget t (initially $t = T$) drops to 0. For a bucket at a position i in a shard s, we consider all *unselected* buckets (s, j) for $j \leq i$, and denote the sum of their costs as C. If $C \leq t$, then we update the cost $t \leftarrow t - C$ and mark all buckets (s, j), $j \leq i$ as selected.

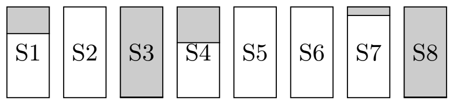

Fig. 1. An illustration of order-aware shard selection. Based on the predicted bucket ranking, portions of shards are selected (shaded area). While strongly relevant shards (S3, S8) may be fully processed, others could terminate early using GRC (S1, S4, S7), or be ignored entirely (S2, S5, S6).

4 Methodology

Data Sets. We conduct our experiments on GOV2 and Clueweb09-B collections, consisting of approximately 25 million and 50 million documents, respectively. Following previous work [9], we remove the documents with Waterloo spam score [8] below 50 from Clueweb09-B. The resulting collection consists of nearly 38 million documents. We use the default HTML parser from the BUbiNG library, and stem terms using the Porter2 algorithm. No stopwords are removed. We use 150 queries from the TREC 04-05 Terabyte Track (GOV2) and 200 queries from the TREC 09-12 Web Track (Clueweb09-B) topics for training our model. We evaluate our method using 10-fold cross-validation.

Training Model. Following [9], we use the SVM$^{\text{rank}}$ library [14] with linear kernel to train our ranking prediction model. It implements the pair-wise approach to learning to rank [18]. We compute the *relevance-based* ground truth as described in [9].

Evaluation Metrics. Following [9], we report search accuracy in terms of P@10 and the more recall-oriented MAP@1000. (Due to space restrictions, we exclude NDCG@30, as it exhibited similar behavior to MAP@1000.) Query cost is reported in terms of the number of postings as described in Section 3.

Search Architecture. The GOV2 collection is clustered into 199 shards, and Clueweb09-B into 123 shards, as done in previous work [9, 10]. We process our queries using the MG4J search engine [5] with default BM25 scoring ($k_1 = 1.2$, $b = 0.5$). When scoring documents in shards, we use global values for collection size, posting count, term frequencies, and occurrence counts, in order to obtain scores that are comparable between shards.

5 Experiments

We experimented with $b = 1, 10, 20$, where $b = 1$ is equivalent to the current state of the art without using a global ordering in [9], which serves as the baseline. Since the baseline disregards differences in query cost due to inverted list lengths during shard selection, we also do so, using the uniform cost model for this step. Following previous work, we report the resulting posting costs in the evaluation.

We express budgets in terms of the numbers of selected shards, where one shard is translated into b buckets for $b > 1$. We report results for budgets up to

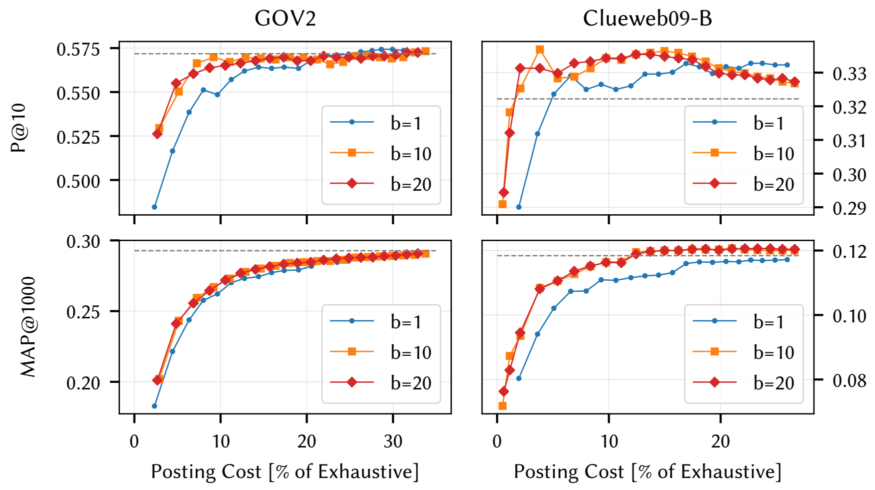

Fig. 2. Quality-cost trade-off of bucket selection for different bucket counts and data sets. Shards were selected under the uniform cost model and with budgets from 1 to 20 shards. For Clueweb09-B, additional budgets of 0.25 and 0.5 were tested for $b > 1$. The quality of exhaustive search is indicated by a dashed line.

20 shards. However, our goal is to limit the cost with little decrease in quality; therefore, we are mostly interested in very low costs, below what has already been shown to work on a par with exhaustive search. Figure 2 shows the trade-offs between quality measures (P@10 and MAP@1000) and query cost in terms of processed postings. Reported values are averaged over all queries.

Both measures improve much faster for B_{10} and B_{20} than for B_1. For instance, P@10 in Clueweb09-B improves upon exhaustive search for a budget as small as 1 shard. At low budgets, we need to process only about a third of what is needed in B_1 to achieve the same quality under P@10. Similarly, achieving the MAP@1000 quality level of exhaustive search requires much fewer postings to be traversed, with smaller improvements observed for GOV2. These improvements are achieved by enabling partial shard access; instead of traversing unimportant documents in highly relevant shards, we allocate budget towards traversing highly important documents in slightly less topically relevant shards. This is achieved by learning of a model that automatically adjusts the cutoff levels.

Shard Access Overhead. While our results show that we can decrease resource requirements, there is an important caveat. One expected side effect of the proposed solution is that for the average query, we contact more shards than for $b = 1$. However, dispatching a query to more shards and collecting the results is likely to result in some amount of overhead. We now address this concern.

As shown in Figure 3, B_{10} and B_{20} contact about 2 to 3 times as many shards as B_1. We now estimate how this would impact the performance of our measures. To do this, we add an overhead cost for contacting a shard equal to 10% of the average cost of an exhaustive query on a single shard. We believe this is a con-

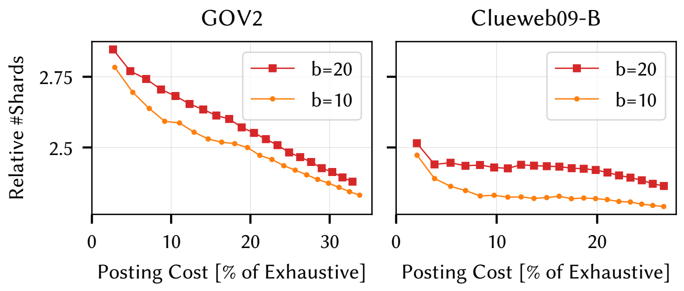

Fig. 3. Average number of shards used in query processing, relative to the baseline.

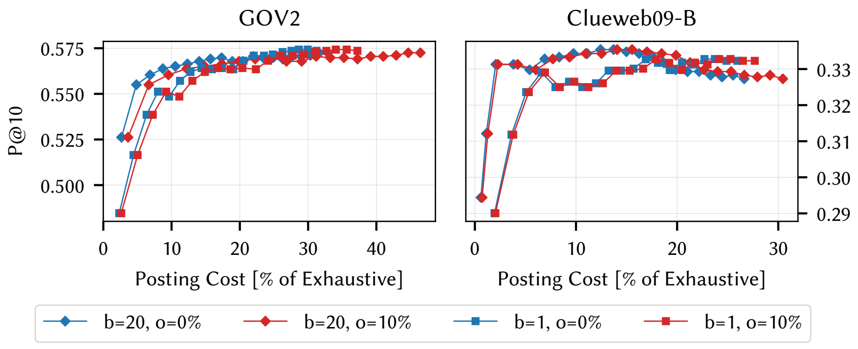

Fig. 4. Quality-cost trade-off of bucket selection with per-shard access overheads of 0% and 10% of the average cost of an exhaustive query on a single shard.

servative upper bound to the overhead of contacting a shard. Figure 4 compares the performance of the no-overhead case (in red) to the 10%-overhead case (in blue), and shows that the improvement over the baseline is still significant.

6 Conclusions

In this paper, we have described how to combine selective search with global ordering-based early termination. Our approach can significantly improve throughput in scenarios where resources are scarce, or where a system experiences peak loads. Although our solution results in additional overhead for contacting more shards, we showed that overall costs still decrease significantly. Our results motivates a number of research questions that we are currently pursuing, including the use of better global orderings based on machine learning, use of additional bucket-level features, the performance of the schemes when used to generate candidates for reranking under a complex ranker, and possible schemes for adapting to high loads under realistic service level agreements (SLAs).

Acknowledgement. This research was partially supported by NSF Grant IIS-1718680 and a grant from Amazon.

References

1. Aly, R., Hiemstra, D., Demeester, T.: Taily: Shard selection using the tail of score distributions. In: Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 673–682 (2013)
2. Anagnostopoulos, A., Becchetti, L., Leonardi, S., Mele, I., Sankowski, P.: Stochastic query covering. In: Proceedings of the 4th ACM International Conference on Web Search and Data Mining, pp. 725–734 (2011)
3. Asadi, N., Lin, J.: Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In: Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 997–1000 (2013)
4. Baeza-Yates, R., Murdock, V., Hauff, C.: Efficiency trade-offs in two-tier web search systems. In: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 163–170. ACM (2009)
5. Boldi, P., Vigna, S.: MG4J at TREC 2005. In: The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings (2005)
6. Broder, A.Z., Carmel, D., Herscovici, M., Soffer, A., Zien, J.: Efficient query evaluation using a two-level retrieval process. In: Proceedings of the 12th International Conference on Information and Knowledge Management, pp. 426–434 (2003)
7. Callan, J.P., Lu, Z., Croft, W.B.: Searching distributed collections with inference networks. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 21–28 (1995)
8. Cormack, G.V., Smucker, M.D., Clarke, C.L.A.: Efficient and effective spam filtering and re-ranking for large web datasets. Information Retrieval **14**(5), 441–465 (2011)
9. Dai, Z., Kim, Y., Callan, J.: Learning to rank resources. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 837–840 (2017)
10. Dai, Z., Xiong, C., Callan, J.: Query-biased partitioning for selective search. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 1119–1128 (2016)
11. Ding, S., Suel, T.: Faster top-k document retrieval using block-max indexes. In: Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 993–1002 (2011)
12. Garcia, S., Williams, H.E., Cannane, A.: Access-ordered indexes. In: Proceedings of the 27th Australasian Conference on Computer Science, pp. 7–14 (2004)
13. Hong, D., Si, L., Bracke, P., Witt, M., Juchcinski, T.: A joint probabilistic classification model for resource selection. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 98–105 (2010)
14. Joachims, T.: Training linear svms in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–226 (2006)
15. Kim, Y., Callan, J., Culpepper, J.S., Moffat, A.: Does selective search benefit from wand optimization? In: Advances in Information Retrieval, pp. 145–158 (2016)
16. Kulkarni, A., Callan, J.: Selective search: Efficient and effective search of large textual collections. ACM Transactions on Information Systems (TOIS) **33**(4), 17 (2015)

17. Leung, G., Quadrianto, N., Tsioutsiouliklis, K., Smola, A.J.: Optimal web-scale tiering as a flow problem. In: Advances in Neural Information Processing Systems, pp. 1333–1341 (2010)
18. Liu, T.Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval **3**(3), 225–331 (2009)
19. Ntoulas, A., Cho, J.: Pruning policies for two-tiered inverted index with correctness guarantee. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 191–198 (2007)
20. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66 (1999)
21. Panigrahi, D., Gollapudi, S.: Document selection for tiered indexing in commerce search. In: Proceedings of the 6th ACM International Conference on Web Search and Data Mining, pp. 73–82. ACM (2013)
22. Richardson, M., Prakash, A., Brill, E.: Beyond pagerank: machine learning for static ranking. In: Proceedings of the 15th International Conference on World Wide Web, pp. 707–715 (2006)
23. Risvik, K.M., Aasheim, Y., Lidal, M.: Multi-tier architecture for web search engines. In: Proceedings of the First Conference on Latin American Web Congress, pp. 132–143 (2003)
24. Si, L., Callan, J.: Relevant document distribution estimation method for resource selection. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 298–305 (2003)
25. Thomas, P., Shokouhi, M.: Sushi: scoring scaled samples for server selection. In: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 419–426 (2009)
26. Turtle, H., Flood, J.: Query evaluation: Strategies and optimizations. Information Processing & Management **31**(6), 831–850 (1995)