Does Selective Search Benefit from WAND Optimization?

Yubin Kim[1(✉)], Jamie Callan[1], J. Shane Culpepper[2], and Alistair Moffat[3]

[1] Carnegie Mellon University, Pittsburgh, USA
yubink@cmu.edu
[2] RMIT University, Melbourne, Australia
[3] The University of Melbourne, Melbourne, Australia

Abstract. Selective search is a distributed retrieval technique that reduces the computational cost of large-scale information retrieval. By partitioning the collection into topical shards, and using a resource selection algorithm to identify a subset of shards to search, selective search allows retrieval effectiveness to be maintained while evaluating fewer postings, often resulting in 90+% reductions in querying cost. However, there has been only limited attention given to the interaction between dynamic pruning algorithms and topical index shards. We demonstrate that the WAND dynamic pruning algorithm is more effective on topical index shards than it is on randomly-organized index shards, and that the savings generated by selective search and WAND are additive. We also compare two methods for applying WAND to topical shards: searching each shard with a separate top-k heap and threshold; and sequentially passing a shared top-k heap and threshold from one shard to the next, in the order established by a resource selection mechanism. Separate top-k heaps provide low query latency, whereas a shared top-k heap provides higher throughput.

Keywords: Selective search · Distributed search · Dynamic pruning · Efficiency

1 Introduction

Selective search is a technique for large-scale distributed search in which the document corpus is partitioned into p topic-based shards during indexing. When a query is received, a resource selection algorithm such as Taily [1] or Rank-S [13] selects the most relevant k shards to search, where $k \ll p$. Results lists from those shards are merged to form a final answer listing to be returned to the user. Selective search has substantially lower computational costs than partitioning the corpus randomly and searching all index shards, which is the most common approach to distributed search [11,12].

Dynamic pruning algorithms such as *Weighted AND* (WAND) [3] and *term-bounded max score* (TBMS) [22] improve the computational efficiency of retrieval systems by eliminating or early-terminating score calculations for documents

N. Ferro et al. (Eds.): ECIR 2016, LNCS 9626, pp. 145–158, 2016.
DOI: 10.1007/978-3-319-30671-1_11

which cannot appear in the top-k of the final ranked list. But topic-based partitioning and resource selection change the environment in which dynamic pruning is performed, and query term posting lists are likely to be longer in shards selected by the resource selection algorithm than in shards that are not selected. As well, each topic-based shard should contain similar documents, meaning that it might be difficult for dynamic pruning to distinguish amongst them using only partial score calculations. Conversely, the documents in the shards that were *not* selected for search might be the ones that a dynamic pruning algorithm would have bypassed if it had encountered them. That is, while the behavior of dynamic pruning algorithms on randomly-organized shards is well-understood, the interaction between dynamic pruning and selective search is not. As an extreme position, it might be argued that selective search is simply achieving the same computational savings that dynamic pruning would have produced, but incurs the additional overhead of clustering the collection and creating the shards. To address these concerns, we investigate the behavior of the well-known *Weighted AND* (WAND) dynamic pruning algorithm in the context of selective search, considering two research questions:

RQ1: Does dynamic pruning improve selective search, and if so, why?
RQ2: Can the efficiency of selective search be improved further using a cascaded pruning threshold during shard search?

2 Related Work

Selective search is a cluster-based retrieval technique [6,19] that combines ideas from conventional distributed search and federated search [12]. Modern cluster-based systems use inverted indexes to store clusters that were defined using criteria such as broad topics [4] or geography [5]. The shards' vocabularies are assumed to be random and queries are sent to a single best shard, forwarding to additional shards as needed [5].

In selective search, the corpus is automatically clustered into query-independent *topic-based shards* with skewed vocabularies and distributed across resources. When a query arrives, a resource selection algorithm identifies a subset of shards that are likely to contain the relevant documents. The selected shards are searched in parallel, and their top-k lists merged to form a final answer. Because only a few shards are searched for each query, total cost per query is reduced, leading to higher throughput.

Previous studies showed that selective search accuracy is comparable to a typical distributed search architecture, but that efficiency is better [1,12], where computational cost is determined by counting the number of postings processed [1,12], or by measuring the execution time of a proof-of-concept implementation.

Resource Selection. Choosing which index shards to search for a query is critical to search accuracy. There are three broad categories of resource selection algorithm: term-based, sample-based, and classification-based. Term-based algorithms model the language distribution of a shard to estimate the relevance

of the shard to a query, with the vocabulary of each shard typically treated as a bag of words. The estimation of relevance is accomplished by adapting an existing document scoring algorithm [8] or by developing a new algorithm specifically for resource selection [1,9,15,24]. Taily [1] is one of the more successful approaches, and fits a Gamma distribution over the relevance scores for each term. At query time, these distributions are used to estimate the number of highly scoring documents in the shard.

Sample-based algorithms extract a small (of the order of 1%) sample of the entire collection, and index it. When a query is received, the sample index is searched and each top-ranked document acts as a (possibly weighted) vote for the corresponding index shard [13,16,20,21,23]. One example is Rank-S [13], which uses an exponentially decaying voting function derived from the document's retrieval rank. The (usually small number of) resources with scores greater than 0.0001 are selected.

Classification-based algorithms use training data to learn models for resources using features such as text, the scores of term-based and sample-based algorithms, and query similarity to historical query logs [2,10]. While classification-based algorithms can be more effective than unsupervised methods, they require access to training data. Their main advantage lies in combining heterogeneous resources such as search verticals.

The Rank-S [13] and Taily [1] have both been used in prior work with similar effectiveness. However Taily is more efficient, because lookups for Gamma parameters are substantially faster than searching a sample index. We use both in our experiments.

Dynamic Pruning. *Weighted AND* (WAND) is a dynamic pruning algorithm that only scores documents that may become one of the current top k based on a preliminary estimate [3]. Dimopoulos et al. [7] developed a Block-Max version of WAND in which continuous segments of postings data are bypassed under some circumstances by using an index where each block of postings has a local maximum score. Petri et al. [17] explored the relationship between WAND-style pruning and document similarity formulations. They found that WAND is more sensitive than Block-Max WAND to the document ranking algorithm. If the distribution of scores is skewed, as is common with BM25, then WAND alone is sufficient. However, if the scoring regime is derived from a language model, then the distribution of scores is top-heavy, and BlockMax WAND should be used. Rojas et al. [18] presented a method to improve performance of systems combining WAND and a distributed architecture with random shards.

Term-Bounded Max Score (TBMS) [22] is an alternative document-at-a-time dynamic pruning algorithm that is currently used in the Indri Search Engine. The key idea of TBMS is to precompute a "topdoc" list for each term, ordered by the frequency of the term in the document, and divided by the document length. The algorithm uses the union of the topdoc lists for the terms to determine a candidate list of documents to be scored. The number of documents in the topdoc list for each term is experimentally determined, a choice that can have an impact on overall performance. Kulkarni and Callan [12] explored the

effects of TBMS on selective search and traditional distributed search architectures. Based on a small set of queries they measured efficiency improvements of 23–40 % for a traditional distributed search architecture, and 19–32 % for selective search, indicating that pruning can improve the efficiency of both approaches.

3 Experiments

The observations of Kulkarni and Callan [12] provide evidence that dynamic pruning and selective search can be complementary. Our work extends that exploration in several important directions. First, we investigate whether there is a correlation between the rank of a shard and dynamic pruning effectiveness for that shard. A correlation could imply that dynamic pruning effectiveness depends on the number of shards searched. We focus on the widely-used WAND pruning algorithm, chosen because it is both efficient and versatile, particularly when combined with a scoring function such as BM25 that gives rise to skewed score distributions [7,17].

Experiments were conducted using the ClueWeb09 Category B dataset, containing 50 million web documents. The dataset was partitioned into 100 topical shards using k-means clustering and a KL-divergence similarity metric, as described by Kulkarni and Callan [11], and stopped using the default Indri stoplist and stemmed using the Krovetz stemmer. On average, the topical shards contain around 500k documents, with considerable variation, see Fig. 1. A second partition of 100 random shards was also created, a system in which exhaustive "all shards" search is the only way of obtaining effective retrieval. Each shard in the two systems was searched using BM25, with $k_1 = 0.9$, $b = 0.4$, and global corpus statistics for *idf* and average document length.[1]

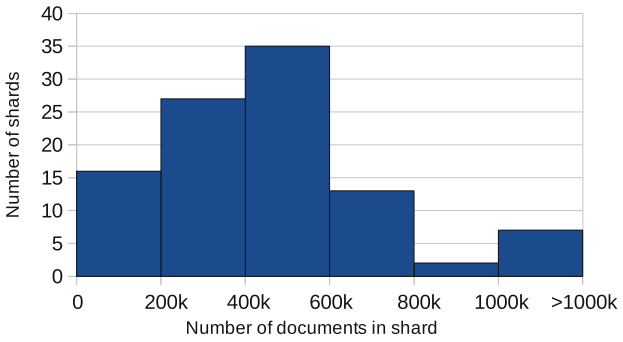


Fig. 1. Distribution of shard sizes, with a total of 100 shards.

[1] The values for b and k_1 are based on the parameter choices reported for Atire and Lucene in the 2015 IR-Reproducibility Challenge, see http://github.com/lintool/IR-Reproducibility.

Each selected shard returned its top 1,000 documents, which were merged by score to produce a final list of $k = 1{,}000$ documents. In selective search, deeper ranks are necessary because most of the good documents may be in one or two shards due to the term skew. Also, deeper k supports learning-to-rank algorithms. Postings lists were compressed and stored in blocks of 128 entries using the FastPFOR library [14], supporting fast block-based skipping during the WAND traversal.

Two resource selection algorithms were used: Taily [1] and Rank-S [13]. The Taily parameters were taken from Aly et al. [1]: $n = 400$ and $v = 50$, where v is the cut-off score and n represents the theoretical depth of the ranked list. The Rank-S parameters used are consistent with the values reported by Kulkarni et al. [13]. A decay base of $B = 5$ with a centralized sample index (CSI) containing 1% of the documents was used – approximately the same size as the average shard. We were unable to find parameters that consistently yielded better results than the original published values.

We conducted evaluations using the first 1,000 unique queries from each of the AOL query log[2] and the TREC 2009 Million Query Track. We removed single-term queries, which do not benefit from WAND, and queries where the resource selection process did not select any shards. Removing single-term queries is a common procedure for research with WAND [3] and allows our results to be compared with prior work. That left 713 queries from the AOL log, and 756 queries from MQT, a total of 1,469 queries.

Our focus is on the efficiency of shard search, rather than resource selection. To compare the efficiency of different shard search methods, we count the num-

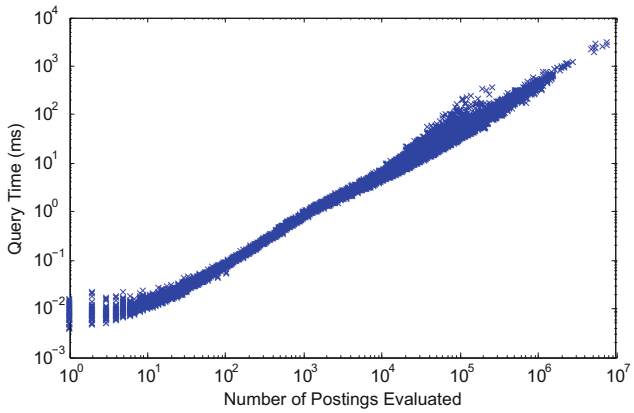


Fig. 2. Correlation between the number of postings processed for a query and the time taken for query evaluation. Data points are generated from MQT queries using both WAND and full evaluation, applied independently to all 100 topical shards and all 100 random shards. In total, $756 \times 200 \times 2 \approx 300{,}000$ points are plotted.

[2] We recognize that the AOL log has been withdrawn, but also note that it continues to be widely used for research purposes.

ber of postings scored, a metric that is strongly correlated with total processing time [3], and is less sensitive to system-specific tuning and precise hardware configuration than is measured execution time. As a verification of this relationship, Fig. 2 shows the correlation between processing time per query, per shard, and the number of postings evaluated. There is a strong linear relationship; note also that more than 99.9 % of queries completed in under 1 s with only a few extreme outliers requiring longer.

Pruning Effectiveness of WAND on Topical Shards. The first experiment investigated how WAND performs on the topical shards constructed by selective search. Each shard was searched independently, as is typical in distributed settings – parallelism is crucial to low response latency. w, the number of posting evaluations required in each shard by WAND-based query evaluation was recorded. The total length of the postings for the query terms in the selected shards was also recorded, and is denoted as b, representing the number of postings processed by an unpruned search in the same shard. The ratio w/b then measures the fraction of the work WAND carried out compared to an unpruned search. The lower the ratio, the greater the savings. Values of w/b can then be combined across queries in two different ways: micro- and macro-averaging. In micro-averaging, w and b are summed over the queries and a single value of w/b is calculated from the two sums. In macro-averaging, w/b is calculated for each query, and averaged across queries. The variance inherent in queries means that the two averaging methods can produce different values, although broad trends are typically consistent.

Figure 3 and Table 1 provide insights into the behavior of macro- and micro-averaging. Figure 3 uses the AOL queries and all 100 topical shards, plotting w/b values on a per query per shard basis as a function of the total length of the postings lists for that query in that shard. Queries involving only rare terms benefit much less from WAND than queries with common terms. Thus, the

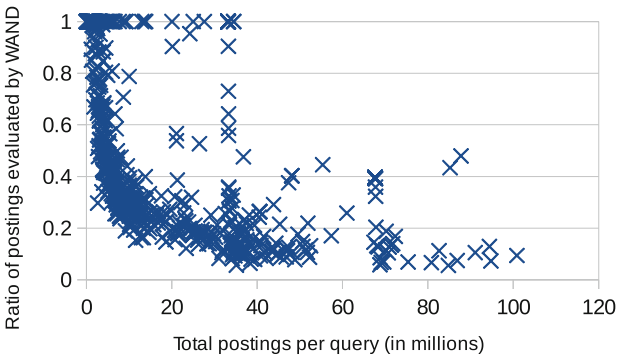


Fig. 3. Ratio of savings achieved by WAND as a function of the total postings length of each query in the AOL set, measured on a per shard basis. A total of $100 \times 713 \approx 71{,}000$ points are plotted. Queries containing only rare terms derive little benefit from WAND.

Table 1. Ratio of per shard per query postings evaluated and per shard per query execution time for WAND-based search, as ratios relative to unpruned search, averaged over 100 topical shards and over 100 randomized shards, and over two groups each of 700+ queries. The differences between the Topical and Random macro-averaged ratios are significant for both query sets and both measures (paired two-tailed t-test, $p < 0.01$).

		WAND postings cost ratio		WAND runtime cost ratio	
		Topical shards	Random shards	Topical shards	Random shards
AOL	micro-averaged	0.35	0.34	0.36	0.38
MQT	micro-averaged	0.36	0.36	0.39	0.43
AOL	macro-averaged	0.51	0.52	0.51	0.53
MQT	macro-averaged	0.60	0.63	0.58	0.63

macro-average of w/b is higher than the micro-average. Micro-averaging more accurately represents the total system savings, whereas macro-averaging allows paired significance testing. We report both metrics in Table 1. The second pair of columns gives millisecond equivalents of w/b, to further validate the postings-cost metric. These values are micro- and macro-averaged w_t/b_t ratios, where w_t is the time in milliseconds taken to process one of the queries on one of the 100 shards using WAND, and b_t is the time taken to process the same query with a full, unpruned search. A key result of Table 1 is that WAND is just as effective across the full set of topical shards as it is on the full set of randomly formed shards. Moreover, the broad trend of the postings cost ratios – that WAND avoids nearly half of the postings – is supported by the execution time measurements.

WAND and Resource Ranking Interactions. The second experiment compares the effectiveness of the WAND algorithm on the shards that the resource ranking algorithm would, and would not, select in connection with each query. The Taily and Rank-S resource selection algorithms were used to determine

Table 2. Average number of shards searched, and micro-averaged postings ratios for those selected shards and for the complement set of shards, together with the corresponding query time cost ratios, in each case comparing WAND-based search to unpruned search. Smaller numbers indicate greater savings.

	Shards searched	WAND postings cost ratio		WAND runtime cost ratio	
		Selected	Non-selected	Selected	Non-selected
Taily AOL	3.1	0.32	0.35	0.36	0.36
Taily MQT	2.7	0.23	0.37	0.30	0.40
Rank-S AOL	3.8	0.27	0.36	0.30	0.37
Rank-S MQT	3.9	0.24	0.37	0.30	0.40

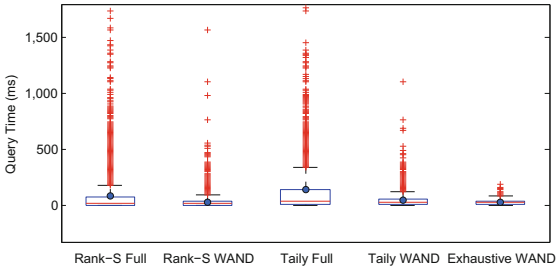
Table 3. As for Table 2, but showing macro-averaged ratios. All differences between selected and non-selected shards are significant (paired two-tailed t-test, $p < 0.01$).

	WAND postings cost ratio		WAND runtime cost ratio	
	Selected	Non-selected	Selected	Non-selected
Taily AOL	0.42	0.52	0.45	0.52
Taily MQT	0.52	0.61	0.53	0.59
Rank-S AOL	0.42	0.53	0.44	0.52
Rank-S MQT	0.52	0.61	0.53	0.60

which shards to search. For each query the WAND savings were calculated for the small set of selected shards, and the much larger set of non-selected shards.

Table 2 lists micro-averaged w/b ratios, and Table 3 the corresponding macro-averaged ratios. While all shards see improvements with WAND, the selected shards see a greater efficiency gain than the non-selected shards, reinforcing our contention that resource selection is an important component in search efficiency. When compared to the ratios shown in Table 1, the selected shards see substantially higher benefit than average shards; the two orthogonal optimizations generate better-than-additive savings.

Figure 4a shows the distribution of the individual per query per shard times for the MQT query set, covering in the first four cases only the shards chosen by the two resource selection processes. The fifth exhaustive search configuration includes data for all of the 100 randomly-generated shards making up the second system, and is provided as a reference point. Figure 4b gives numeric values for



(a)

	Mean	Median
Rank-S Full	85.0	13.0
Rank-S WAND	28.5	11.3
Taily Full	134.0	34.2
Taily WAND	42.7	23.6
Exhaustive WAND	26.6	21.8

(b)

Fig. 4. Distribution of query response times for MQT queries on shards: (a) as a box plot distribution, with a data point plotted for each query-shard pair; (b) as a table of corresponding means and medians. In (a), the center line of the box indicates the median, the outer edges of the box the first and third quartiles, and the blue circle the mean. The whiskers extend to include all points within 1.5 times the inter-quartile range of the box. The graph was truncated to omit a small number of extreme points for both Rank-S Full and Taily-Full. The maximum time for both these two runs was 6,611 ms.

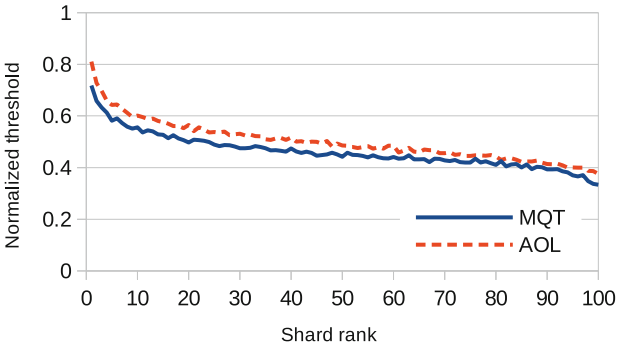


Fig. 5. Normalized 1,000 th document scores from shards, averaged over queries and then shard ranks, and expressed as a fraction of the collection-wide maximum document score for each corresponding query. The score falls with rank, as fewer high-scoring documents appear in lower-ranked shards.

the mean and median of each of the five distributions. When WAND is combined with selective search, it both reduces the average time required to search a shard and also reduces the variance of the query costs. Note the large differences between the mean and median query processing times for the unpruned search and the reduction in that gap when WAND is used; this gain arises because query and shard combinations that have high processing times due to long postings lists are the ones that benefit most from WAND. Therefore, in typical distributed environments where shards are evaulated in parallel, the slowest, bottleneck shard will benefit the most from WAND and may result in additional gains in latency reduction. Furthermore, while Fig. 4 shows similar per shard query costs for selective and exhaustive search, the total *work* associated with selective search is substantially less than exhaustive search because only 3–5 shards are searched per query, whereas exhaustive search involves all 100 shards. Taken in conjunction with the previous tables, Fig. 4 provides clear evidence that WAND amplifies the savings generated by selective search, answering the first part of RQ1 with a "yes". In addition, these experiments have confirmed that execution time is closely correlated with measured posting evaluations. The remaining experiments utilize postings counts as the cost metric.

We now consider the second part of RQ1 and seek to explain *why* dynamic pruning improves selective search. Part of the reason is that the postings lists of the query terms associated with the highly ranked shards are longer than they are in a typical randomized shard. With these long postings lists, there is more opportunity for WAND to achieve early termination. Figure 5 shows normalized final heap-entry thresholds, or equivalently, the similarity score of the 1,000 th ranked document in each shard. The scores are expressed as a fraction of the maximum document score for that query across all shards, then plotted as a function of the resource selector's shard ranking using Taily, averaged over queries. Shards that Taily did not score because they did not contain any query

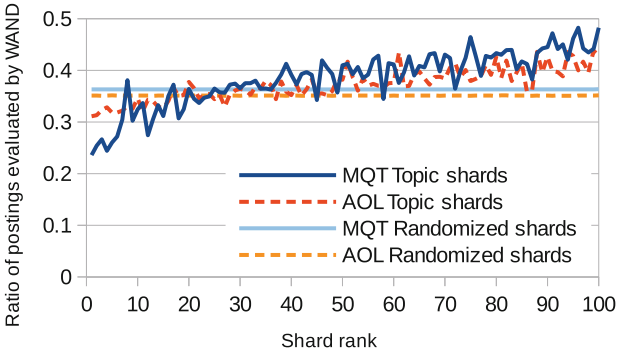


Fig. 6. The micro-average w/b ratio for WAND postings evaluations, as a function of the per query shard rankings assigned by Taily. Early shards generate greater savings.

terms were ordered randomly. For example, for the AOL log the 1,000 th document in the shard ranked highest by Taily attains, on average across queries, a score that is a little over 80 % of the maximum score attained by any single document for that same query. The downward trend in Fig. 5 indicates that the resource ranking process is effective, with the high heap-entry thresholds in the early shards suggesting – as we would hope – that they contain more of the high-scoring documents.

To further illustrate the positive relationship between shard ranking and WAND, w/b was calculated for each shard in the per query shard orderings, and then micro-averaged at each shard rank. Figure 6 plots the average as a function of shard rank, and confirms the bias towards greater savings on the early shards – exactly the ones selected for evaluation. As a reference point, the same statistic was calculated for a random ordering of the randomized shards (random since no shard ranking is applied in traditional distributed search), with the savings ratio being a near-horizontal line. If an unpruned full search were to be plotted, it would be a horizontal line at 1.0. The importance of resource selection to retrieval effectiveness has long been known; Fig. 6 indicates that effective resource selection can improve overall efficiency as well.

Improving Efficiency with Cascaded Pruning Thresholds. In the experiments reported so far, the rankings were computed on each shard independently, presuming that they would be executing in parallel and employing private top-k heaps and private heap-entry thresholds, with no ability to share information. This approach minimizes search latency when multiple machines are available, and is the typical configuration in a distributed search architecture. An alternative approach is suggested by our second research question: what happens if the shards are instead searched sequentially, passing the score threshold and top-k heap from each shard to the next? The heap-entry score threshold is then non-decreasing across the shards, and additional savings should result. While this approach would be unlikely to be used in an on-line system, it provides

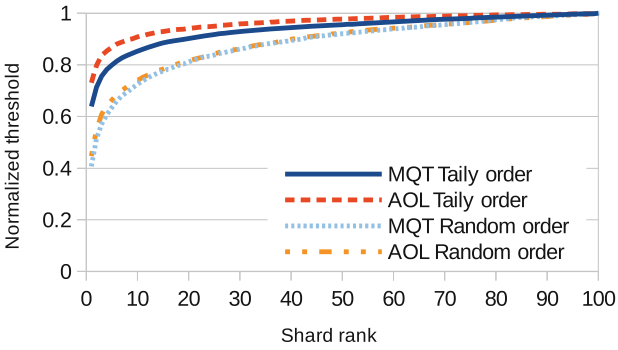


Fig. 7. Normalized 1,000 th document scores from shards relative to the highest score attained by any document for the corresponding query, micro-averaged over queries, assuming that shards are processed sequentially rather than in parallel, using the Taily-based ordering of topical shards and a random ordering of the same shards.

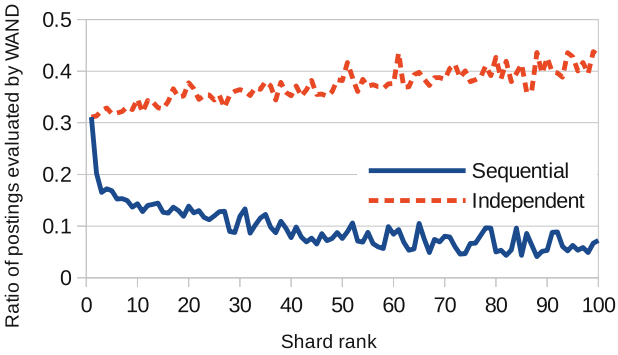


Fig. 8. Ratio of postings evaluated by WAND for independent shard search versus sequential shard search, AOL queries with micro-averaging. Shard ranking was determined by Taily.

an upper bound on the efficiency gains that are possible if a single heap was shared by all shards, and would increase throughput when limited resources are available and latency is not a concern: for example, in off-line search and text analytics applications.

Figure 7 demonstrates the threshold in the sequential WAND configuration, with shards ordered in two ways: by Taily score, and randomly. The normalized threshold rises quickly towards the maximum document score through the first few shards in the Taily ordering, which is where most of the documents related to the query are expected to reside. Figure 8 similarly plots the w/b WAND savings ratio at each shard rank, also micro-averaged over queries, and with shard ordering again determined by the Taily score. The independent and sequential configurations diverge markedly in their behavior, with a deep search in the latter processing far fewer postings than a deep search in the former. The MQT

query set displayed similar trends. Sharing the dynamic pruning thresholds has a large effect on the efficiency of selective search.

Our measurements suggest that a hybrid approach between independent and sequential search could be beneficial. A resource-ranker might be configured to underestimate the number of shards that are required, with the understanding that a second round of shard ranking can be instigated in situations where deeper search is needed, identified through examining the scores or the quantity of documents retrieved. When a second wave of shards is activated, passing the maximum heap-entry threshold attained by the first-wave process would reduce the computational cost. If the majority of queries are handled within the first wave, a new combination of latency and workload will result.

4 Conclusion

Selective search reduces the computational costs of large-scale search by evaluating fewer postings than the standard distributed architecture, resulting in computational work savings of up to 90 %. To date there has been only limited consideration of the interaction between dynamic pruning and selective search [12], and it has been unclear whether dynamic pruning methods improve selective search, or whether selective search is capturing some or all of the same underlying savings as pruning does, just via a different approach. In this paper we have explored WAND dynamic pruning using a large dataset and two different query sets. In contrast to Kulkarni's findings with TBMS [12], we show that WAND-based evaluation and selective search generate what are effectively independent savings, and that the combination is more potent than either technique is alone – that is, that their interaction is a positive one. In particular, when resource selection is used to choose query-appropriate shards, the improvements from WAND on the selected shards is greater than the savings accruing on random shards, confirming that dynamic pruning further improves selective search – a rare situation where orthogonal optimizations are better-than-additive. We also demonstrated that there is a direct correlation between the efficiency gains generated by WAND and the shard's ranking. While it is well-known that resource selection improves effectiveness, our results suggest that it can also improve overall efficiency too.

Finally, two different methods of applying WAND to selective search were compared and we found that passing the top-k heap through a sequential shard evaluation greatly reduced the volume of postings evaluated by WAND. The significant difference in efficiency between this approach and the usual fully-parallel mechanism suggests avenues for future development in which hybrid models are used to balance latency and throughput in novel ways.

Acknowledgments. This research was supported by National Science Foundation (NSF) grant IIS-1302206; a Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate Scholarship-Doctoral award; and the Australian Research Council (ARC) under the *Discovery Projects* scheme (DP140103256). Shane Culpepper is the recipient of an Australian Research Council (ARC) DECRA Research Fellowship (DE140100275).

References

1. Aly, R., Hiemstra, D., Demeester, T.: Taily: shard selection using the tail of score distributions. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 673–682 (2013)
2. Arguello, J., Callan, J., Diaz, F.: Classification-based resource selection. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 1277–1286 (2009)
3. 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)
4. Cacheda, F., Carneiro, V., Plachouras, V., Ounis, I.: Performance comparison of clustered and replicated information retrieval systems. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECIR 2007. LNCS, vol. 4425, pp. 124–135. Springer, Heidelberg (2007)
5. Cambazoglu, B.B., Varol, E., Kayaaslan, E., Aykanat, C., Baeza-Yates, R.: Query forwarding in geographically distributed search engines. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 90–97 (2010)
6. Croft, W.B.: A model of cluster searching based on classification. Inf. Syst. **5**(3), 189–195 (1980)
7. Dimopoulos, C., Nepomnyachiy, S., Suel, T.: Optimizing top-k document retrieval strategies for block-max indexes. In: Proceedings of the of the Sixth ACM International Conference on Web Search and Data Mining, pp. 113–122 (2013)
8. Gravano, L., García-Molina, H., Tomasic, A.: GlOSS: Text-source discovery over the internet. ACM Trans. Database Syst. **24**, 229–264 (1999)
9. Ipeirotis, P.G., Gravano, L.: Distributed search over the hidden web: Hierarchical database sampling and selection. In: Proceedings of the 28th International Conference on Very Large Data Bases, pp. 394–405 (2002)
10. Kang, C., Wang, X., Chang, Y., Tseng, B.: Learning to rank with multi-aspect relevance for vertical search. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, pp. 453–462 (2012)
11. Kulkarni, A., Callan, J.: Document allocation policies for selective searching of distributed indexes. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 449–458 (2010)
12. Kulkarni, A., Callan, J.: Selective search: Efficient and effective search of large textual collections. ACM Trans. Inf. Syst. **33**(4), 17:1–17:33 (2015)
13. Kulkarni, A., Tigelaar, A., Hiemstra, D., Callan, J.: Shard ranking and cutoff estimation for topically partitioned collections. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 555–564 (2012)
14. Lemire, D., Boytsov, L.: Decoding billions of integers per second through vectorization. Soft. Prac. & Exp. **41**(1), 1–29 (2015)
15. Nottelmann, H., Fuhr, N.: Evaluating different methods of estimating retrieval quality for resource selection. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 290–297. ACM (2003)
16. Paltoglou, G., Salampasis, M., Satratzemi, M.: Integral based source selection for uncooperative distributed information retrieval environments. In: Proceedings of the 2008 ACM Workshop on Large-Scale Distributed Systems for Information Retrieval, pp. 67–74 (2008)

17. Petri, M., Culpepper, J.S., Moffat, A.: Exploring the magic of WAND. In: Proceedings of the Australian Document Computing Symposium, pp. 58–65 (2013)
18. Rojas, O., Gil-Costa, V., Marin, M.: Distributing effciently the block-max WAND algorithm. In: Proceedings of the 2013 International Conference on Computational Science, pp. 120–129 (2013)
19. Salton, G.: Automatic Information Organization and Retrieval. McGraw-Hill, New York (1968)
20. Shokouhi, M.: Central-Rank-Based Collection Selection in Uncooperative Distributed Information Retrieval. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECIR 2007. LNCS, vol. 4425, pp. 160–172. Springer, Heidelberg (2007)
21. 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 Informaion Retrieval, pp. 298–305 (2003)
22. Strohman, T., Turtle, H., Croft, W.B.: Optimization strategies for complex queries. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 219–225 (2005)
23. Thomas, P., Shokouhi, M.: Sushi: Scoring scaled samples for server selection. In: Proceedings of the 32nd ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 419–426 (2009)
24. Yuwono, B., Lee, D.L.: Server ranking for distributed text retrieval systems on internet. In: Proceedings of the International Conference on Database Systems for Advanced Applications, pp. 41–49 (1997)