Efficient and Effective Retrieval using Selective Pruning

Nicola Tonellotto
Information Science and Technologies Institute
National Research Council
56124 Pisa, Italy
nicola.tonellotto@isti.cnr.it

Craig Macdonald, Iadh Ounis
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, UK
{craig.macdonald,iadh.ounis}@glasgow.ac.uk

ABSTRACT

Retrieval can be made more efficient by deploying dynamic pruning strategies such as WAND, which do not degrade effectiveness up to a given rank. It is possible to increase the efficiency of such techniques by pruning more 'aggressively'. However, this may reduce effectiveness. In this work, we propose a novel selective framework that determines the appropriate amount of pruning aggressiveness on a per-query basis, thereby increasing overall efficiency without significantly reducing overall effectiveness. We postulate two hypotheses about the queries that should be pruned more aggressively, which generate two approaches within our framework, based on query performance predictors and query efficiency predictors, respectively. We thoroughly experiment to ascertain the efficiency and effectiveness impacts of the proposed approaches, as part of a search engine deploying state-of-the-art learning to rank techniques. Our results on 50 million documents of the TREC ClueWeb09 collection show that by using query efficiency predictors to target inefficient queries, we observe that a 36% reduction in mean response time and a 50% reduction of the response times experienced by the slowest 10% of queries can be achieved while still ensuring effectiveness.

Categories and Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval

Keywords: Efficient & Effective Search Engines, Dynamic Pruning, Learning to Rank, Query Efficiency Prediction

1. INTRODUCTION

Web search engines and other large-scale information retrieval (IR) systems are not just concerned with the quality of search results (also known as *effectiveness*), but also with the speed with which the results are obtained (*efficiency*). These aspects form a natural tradeoff that all search engines must address, in that many approaches that increase effectiveness may have a corresponding impact on efficiency due to their complex nature [35].

Increasingly, search engines deploy learning to rank approaches, whereby a learned model combines many features into an effective approach for ranking [20]. Our work is firmly placed in a learning to rank setting, where a typical search engine consists of three phases of operation, as follows (illustrated in Figure 1):

WSDM'13, February 4–8, 2013, Rome, Italy.

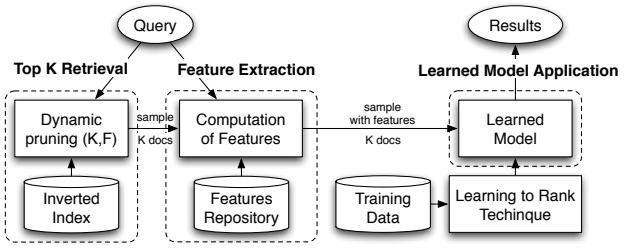


Figure 1: Phases of retrieval by a search engine.

Top K Retrieval: An initial ranking strategy selects K documents from the inverted index, identified using a single feature (often the BM25 weighting model) [7, 20]. Dynamic pruning strategies such as WAND can be applied to efficiently generate the set of K results (called the *sample* [20, 25]).

Feature Extraction: Computation of additional features for each document in the sample, such as other weighting models, including those calculated on different fields (anchor text, title, etc.) or query independent features (URL length, PageRank, etc.).

Learned Model Application: Re-ranking of the sample results by the application of a learned model, obtained by earlier learning on training data, to increase effectiveness compared to the sample [20].

The first phase of the retrieval process – where the K documents of the sample are identified – is data intensive, and hence, as we will later show, has the largest impact on efficiency. For this reason, efficient retrieval strategies such as the WAND dynamic pruning strategy can be deployed. Indeed, WAND can enhance efficiency by avoiding the scoring of documents that can never be retrieved in the top K results, without degrading the effectiveness up to rank K, known as *safe-to-rank-K*. WAND can be made more efficient by reducing the number of documents K to be retrieved. It is possible to further increase the efficiency of WAND by applying the pruning more *aggressively*, but at loss of the guaranteed safeness, with possible degradations in the effectiveness of the results. The Top K Retrieval phase also impacts overall effectiveness in two manners: (i) Decreasing K such that a smaller sample is obtained may miss relevant documents at deeper ranks, which will have no chance of being re-ranked towards the top by the learned model, hence degrading the overall effectiveness of the search engine; (ii) The pruning aggressiveness used to obtain the sample may also impact the resulting effectiveness after re-ranking by the learned model, as the sample is no longer safe-to-rank-K.

In this work, we aim to ensure effective and efficient retrieval, by selecting which queries should be pruned more aggressively. In particular, not all queries have similar effectiveness. Moreover, not all queries have equal response time,

even with dynamic pruning [23] - yet search engine users are not willing to wait long for queries to be answered [33]. Hence, our work addresses the following key question: for which queries should the pruning be applied more aggressively to obtain efficient yet effective retrieval? We examine two novel approaches for selecting queries that can be aggressively pruned: those that are predicted to be easy, and hence can tolerate a lower quality sample; and those that are predicted to be particularly inefficient. We postulate that easier queries have more relevant documents highly ranked within the sample, and hence can be more aggressively pruned. On the other hand, targeting queries that will be inefficient can attain more efficiency gains, but may have more impact on effectiveness. Moreover, as the effectiveness and efficiency of a query cannot be known before retrieval commences, our approaches use pre-retrieval query performance predictors [8] and query efficiency predictors [23], to select an appropriate amount of pruning for each query.

The contributions of this work are as follows: We provide an analysis on how the parameters affecting the aggressiveness of a dynamic pruning strategy impact on the effectiveness of a learned model; Moreover, we propose a selective pruning framework for identifying the appropriate pruning setting of a dynamic pruning strategy on a per-query basis, to ensure efficient yet effective retrieval. Two methods are proposed, based on predicting the effectiveness of the query, or on predicting its efficiency. To the best of our knowledge, this is the first work on the selective application of dynamic pruning on a per-query basis. Experiments are conducted on a standard TREC Web test collection of 50 million documents, deploying two learned models from two learning to rank techniques. Our results show that marked improvements in efficiency can be obtained while ensuring effectiveness, with the strongest results being obtained when the queries that are predicted to be inefficient are selected for aggressive pruning.

The remainder of this paper is structured as follows: Section 2 provides background material on dynamic pruning for efficient retrieval; In Section 3, we review related work that examined both efficiency and effectiveness; Section 4 introduces the proposed framework for selectively determining pruning on a per-query basis; Section 5 details the experimental setup used in this paper; Section 6 analyses the efficiency and effectiveness of different pruning settings for the K and F parameters. Experiments and conclusions follow in Sections 7 & 8, respectively.

2. BACKGROUND

Even in large search engines, the matching of documents in the Top K Retrieval phase still uses the classical inverted index data structure [14], by traversing the postings lists for each query term. Moreover, due to its data intensive nature [23], we will later show that this represents the largest contribution to the time that a search engine takes to retrieve documents in response to a query.

Various techniques have been proposed that improve efficiency by pruning documents that are unlikely to be retrieved. For example, in [3], Anh & Moffat proposed a pruning strategy that uses posting lists pre-sorted by the impact of the postings. Alternatively, documents that are unlikely to be retrieved can be removed from the index [5]. Both approaches can cause possible loss of effectiveness. However, we focus on techniques that can be configured to be *safe-to-rank-K* – i.e., cannot degrade retrieval effectiveness up to a given rank K – and use docid sorted posting lists, since this is deployed by at least one major commercial search engine [14]. In particular, *dynamic pruning* strategies aim to avoid the scoring of postings for documents that cannot make the top K retrieved set.

All state-of-the-art safe dynamic pruning strategies [6, 30, 34][1] aim to avoid scoring parts of the posting lists, to save disk access, decompression and score computation costs. This mechanism is implemented by maintaining additional information during retrieval, namely: a *threshold* τ, which is the minimum score that documents must achieve to have a chance to be present in the final top K results; and, for each query term, a *term upper bound*, which is the maximal contribution of that particular term to any document score. The upper bound is usually obtained by pre-scanning the posting lists of each term at indexing time, to record the maximum score found in each posting list [21].

One such dynamic pruning strategy is WAND [6], which represents the state-of-the-art in safe dynamic pruning [16]. WAND repeatedly calculates a *pivot term*, based on the terms that the next document must contain to exceed threshold τ. The next document containing the pivot term is the *pivot document*, which will be the next document to be fully scored. In this manner, WAND makes use of skipping [28] forward in posting lists, which reduces posting list decompression overheads, and can reduce IO, with resulting improvements in efficiency [16, 21]. Moreover, importantly for our work, the overall tradeoff between efficiency and effectiveness in WAND can be adjusted by changing parameters affecting the threshold and the size of K. A key novelty of our work is showing how these parameters can be automatically selected for each query.

3. RELATED WORK

This paper proposes a selective framework for adjusting the effectiveness and efficiency of WAND on a per-query basis. In particular, we build upon the experiments of Broder et al. [6], who examined how the efficiency and effectiveness of WAND are affected by increasing the *aggressiveness* of the pruning. In particular, two parameters were shown to affect the efficiency of WAND. Firstly, with a smaller K, the threshold τ, i.e., the current minimum score that documents must achieve to have a chance to be present in the final top K results, is higher, and hence more pruning of low-scoring documents can be achieved. Secondly, since a new document is scored only when its approximate score beats the current threshold τ, the threshold can be artificially increased by using a new threshold $\tau' = F \cdot \tau$, where $F \geq 1$ is a tunable parameter. In doing so, the dynamic pruning strategy is not guaranteed to be safe-to-rank-K anymore, but new documents are scored only if they can achieve a score F times greater than the current threshold τ, with resulting improvements in efficiency. The experiments in [6] showed that increasing F and decreasing K did increase efficiency, but that effectiveness was impacted for a sufficiently large F. However, their work predates modern IR, where machine learned models are applied directly on the ranking produced by the dynamic pruning strategy. In contrast, in our work, we not only show the impact of K and F on a Web corpus that is 30 times larger than that used in [6], but we also investigate the impact of the pruning parameters on the effectiveness of a learned model, where the lower ranks of the sample are also

[1]We omit the database-focused threshold algorithms of Fagin et al. [15], which assume lists sorted by score, while we assume docid-based sorting.

important. Moreover, we show how different K and F can be selected on a per-query basis to attain efficient retrieval.

There exists some previous work on selective approaches for information retrieval. For instance, Amati et al. [2] noted that query expansion underperforms for difficult queries. By identifying such difficult queries using *query performance predictors* (QPP), they were able to selectively apply query expansion, i.e. decide whether to apply query expansion on a per-query basis. Similarly, Cronen-Townsend et al. also used query performance predictors for selective query expansion [13]. Plachouras & Ounis [31] applied different retrieval approaches for Web search on a per-query basis. Other recent work tackles the efficiency/effectiveness tradeoff. In particular, Wang et al. [35] devised a tradeoff metric that measures both effectiveness and efficiency. By using this within a learning to rank technique, they could identify uni- and bi-gram query term "features" to discard within the Sequential Dependence model [26], to balance efficiency and effectiveness. However, as their work is not concerned with more common document features, it is less widely applicable.

In a work applicable to document features for learning to rank, Wang et al. [36] observed that the expense of applying a feature was related to the number of documents on which that feature was computed. As only the final top-ranked documents are required, they adapted a learning to rank technique such that it also selects how many low-ranked documents should be discarded before the application of each feature. Hence the output of applying the learned model is an effective ranking shorter than the input sample, with features calculated on a reduced number of documents. In our work, we will show that the initial Top K Retrieval is the most expensive of the three phases. While the approach of [36] places emphasis on reducing the number of documents for feature calculations, we focus on adjusting the efficiency and effectiveness of the Top K Retrieval phase.

Learned models encompassing additive regression trees, such as those obtained from the LambdaMART [37] learning to rank technique, are more complex to apply than those that are simple linear combinations of feature values (e.g. [27]). For this reason, Cambazoglu et al. [7], showed how such regression trees could be pruned at retrieval time - although effectiveness could be slightly degraded. This work is orthogonal to ours, in that it could be used to reduce the duration of the Learned Model Application phase. However, instead of pruning regression trees from the learned model, our work aims to reduce the expense in identifying the documents in the first Top K Retrieval phase.

No work from the literature has examined the full picture concerning the application of dynamic pruning within the context of a learned model, such that the interplay between effectiveness and efficiency is examined. Indeed, in the next section, we propose a framework that selects an appropriate K and F settings for WAND on a per-query basis, ensuring effective yet efficient retrieval. Differing from previous work, this framework varies the aggressiveness of the dynamic pruning strategy deployed in the Top K Retrieval phase on a per-query basis, by making use of efficiency and effectiveness predictions.

4. SELECTIVE PRUNING FRAMEWORK

Dynamic pruning techniques such as WAND permit the efficient retrieval compared to an exhaustive scoring of all documents containing occurrences of every query term. For $F = 1$, the produced ranking is safe-to-rank-K. As mentioned above, WAND can prune documents more aggressively by increasing a factor F on the current threshold τ, or by retrieving less documents (reducing K). Changes to K and F offer efficiency advantages, but can have disadvantages in terms of effectiveness. For instance, if K is decreased, the ranking up to K documents is still safe, however as the sample may have less relevant documents for the learner to identify, the learned model effectiveness can be degraded. On the other hand, while increasing F no longer guarantees that the document ranking will have the same effectiveness as for $F = 1$, Broder et al. [6] showed that the top ranked documents often remain unchanged for small values of $F > 1$ while still permitting efficiency benefits. However, as mentioned in Section 3, they did not consider the impact of pruning on the effectiveness of learned models, which often markedly re-rank documents from deep within the sample.

Our proposed selective pruning framework obtains efficiency benefits by applying aggressive pruning for queries where possible, based on appropriate selections of K and F when generating the sample. In particular, the selection is based upon two hypotheses about which queries can be more aggressively pruned without damaging overall effectiveness. For each of these hypotheses, based on a prediction for each query, a selection of the appropriate K and F values is made. The outline for the selective pruning framework is shown in Algorithm 1: steps 2-4 follow the familiar outline of the three retrieval phases from Figure 1; in step 1, instead of setting K and F uniformly for all queries, the SELECT($\bullet$) function selects appropriate values for each query, taking into account a prediction from PREDICT(q) for query q.

Algorithm 1 Selective Pruning Framework

Input: The query q
Output: The top K ranked documents
1: $\{K, F\} \leftarrow$ SELECT(PREDICT(q))
2: Sample $S \leftarrow$ WAND(q,K,F)
3: Features $T \leftarrow$ EXTRACT(q,S)
4: Results $A \leftarrow$ APPLY(T, MODEL(K,F))

We investigate two approaches – based on predicted effectiveness (Sections 4.1) and efficiency (Section 4.2) – for selecting the settings of pruning parameters K and F for WAND for each query. Each approach uses different instantiations of both SELECT($\bullet$) and PREDICT(q).

4.1 Pruning Effective Queries

Marginally unsafe pruning (e.g. $F = 2$ or 3) by WAND can leave the top-ranked documents unchanged [6]. Moreover, given the usual effectiveness of the weighting model used to generate the sample, relevant documents are expected to appear at high ranks in the sample. However, when the output of WAND is re-ranked as part of a learned model, the learned model may rank highly documents that were originally ranked quite low in the sample. Such lowly-ranked sample documents are more likely to be omitted by an aggressive unsafe setting of WAND, thereby possibly degrading the effectiveness of the learned model. We postulate that some queries are 'easier' than others, where the applied learned model does not have to re-rank relevant documents from deep in the document ranking. In other words, K is unnecessarily large for such queries, and their effectiveness is less likely to be affected by more aggressive pruning. However, it is impossible to know a-priori how large a sample is necessary for effective retrieval for a given query. Instead, we turn to the field of pre-retrieval query performance prediction [8], where various techniques have been developed to predict the effectiveness of a query before retrieval commences. Hence, we arrive at our first hypothesis:

HYPOTHESIS 1. *Queries that are predicted to be easier should be more aggressively pruned to benefit response time while minimising effectiveness degradations.*

Based on this hypothesis, in Section 6, we propose a definition for $\text{SELECT}_{QPP}(\bullet)$ using the predicted effectiveness $\widehat{E}(q)$ of query q, as determined by pre-retrieval query performance predictor(s) [19] (QPP).

4.2 Pruning Inefficient Queries

While efficiency may be improved by applying WAND compared to an exhaustive scoring of all documents containing occurrences of every query term, the response time of a query q (denoted $R(q)$) can markedly differ from another query with similar surface characteristics (e.g. number of query terms, length of posting lists) [23].

Given that some queries are particularly inefficient, they should naturally be targeted for improving efficiency. The key to this intuition is being able to predict the queries that will be inefficient, and hence predict the response time of the query, denoted $\widehat{R}(q)$. Indeed, the amount of pruning possible by WAND for different queries is variable, depending on the distribution of scores among the postings of the constituent query terms [23]. Essentially, the pruning difficulty of a query is related to how quickly the threshold τ rises to exclude documents as retrieval proceeds. Queries for which the high value documents occur more towards the start of the posting lists are more likely to be easier to prune [23]. However, as there is a universe of possible queries with different constituent terms, Macdonald et al. [23] showed that an accurate model for predicting the response time of a given query can be obtained by aggregating statistics for the constituent query terms. These statistics - such as the number of postings, and the number of postings with scores within 5% of the upper bound - can be calculated at indexing time, and hence can be easily obtained as the query arrives, such that the response time for a query can be quickly and accurately predicted before retrieval commences. Hence, in this work, we obtain $\widehat{R}(q)$ from response time estimates from query efficiency predictors (QEP). The predicted response time is then used to select the K and F values before the IO-intensive dynamic pruning phase starts:[2]

HYPOTHESIS 2. *Queries that are predicted to take longer to retrieve should be more aggressively pruned to benefit response time while minimising effectiveness degradations.*

Based on this hypothesis, in Section 6 we propose a definition for $\text{SELECT}_{QEP}(\bullet)$ using the predicted response time of a query $\widehat{R}(q)$. In the following section, we provide the experimental setup, which we use to obtain the empirical analysis, from which the definitions for $\text{SELECT}_{QPP}(\bullet)$ and $\text{SELECT}_{QEP}(\bullet)$ follow.

5. EXPERIMENTAL SETUP

In the following, we describe our experimental setup in terms of the corpus and queries used (Section 5.1), and the three retrieval phases described in Section 1: Top K Retrieval (Section 5.2), Feature Extraction (Section 5.3), and Learned Model Application (Section 5.4). We describe how efficiency is measured in Section 5.5. Finally, the setups for query effectiveness and efficiency predictors are described in Sections 5.6 & 5.7, respectively.

[2]Our experiments have found that an efficiency prediction can be made in a few milliseconds.

5.1 Documents & Queries

The experiments in the following sections are conducted using the large TREC ClueWeb09 category B corpus, which consists of 50 million Web documents. ClueWeb09 provides a larger, more realistic Web setting than other TREC Web corpora, such as GOV2. Indeed, this corpus has both TREC topics with relevance assessments, as well as a generally available query log. As we wish to measure and analyse both efficiency and effectiveness, we use two separate topics sets, for measuring each in separation.

Best practices in efficiency experiments demand a large number of queries, however the number used in the literature can vary widely, from the 50-150 TREC topics used in [35] to thousands of queries from commercial search engines used in [7]. In this work, we use a set of consecutive user queries from a generally available real search engine log, thereby measuring the mean query response time for retrieval. In particular, we select the first 1,000 queries of the MSN 2006 query log [12], without any stemming and without removing standard stopwords. This amounts to 981 queries (queries with no matching documents are discarded). This query set exhibits all of the expected properties of a query log, such as frequently repeated 'head' queries and a tail of infrequent queries. The mean query length is 2.7 terms. For training/testing, the query set is split chronologically: 500 for training, 481 for testing. Finally, we note that the LETOR learning to rank datasets [20] are not suitable for our experiments as they use a safe sample with fixed $K = 1000$.

A query set with relevance assessments is required for measuring effectiveness. For this reason, we use the 150 topics of the TREC 2009-2011 Web tracks, which form a test collection based on ClueWeb09 [10]. In particular, for the purposes of learning to rank, these topics are mixed and split into three equally sized training, validation and test sets. Effectiveness is measured using NDCG@20, an official TREC Web track measure [10].

5.2 Top K Retrieval Phase

We index ClueWeb09 using the Terrier IR platform,[3] applying no stemming, and keeping all stopwords. Term positions are recorded within the compressed inverted index, as well as terms from the URLs and titles of documents, and any incoming anchor text, as separate fields [22]. We also build skip lists for the inverted index [28], with a skip pointer every 1,000 postings. The resulting index size is 126GB.

During the Top K Retrieval phase, the WAND [6] dynamic pruning strategy is used to select K documents, where each document has been scored for each query using the parameter-free DPH Divergence from Randomness weighting model [1]. DPH is a parameter-free model that exhibits similar effectiveness to BM25 without requiring any training [1]. As an effective default setting for WAND, we use the safe $F = 1$, and $K = 1000$. The latter is the sample size used in the LETOR learning to rank v3.0 datasets [20].

5.3 Feature Extraction Phase

We deploy 33 document features in the feature extraction phase of our ranking approach, covering the most typically applied categories of query independent (encompassing URL, link analysis and content quality features) and query dependent features (including term weighting models and term dependence models). Indeed, many of these features have been shown to create effective learned models on the ClueWeb09 corpus [32]. Table 1 lists the deployed features.

[3]`http://terrier.org/`

Type	Features	Total
QD	Weighting models (DPH [1], PL2 [1], BM25, Dirichlet Language Models, Matching Query Terms [32])	21
QD	Fields-based model (PL2F [22])	1
QI	URL/link analysis features (e.g. PageRank, Outlinks)	3
QI	Quality features (e.g. spamminess [11], field lengths, fraction of stopwords, table text [4])	5
QI	Click feature (click count)	1
QD	term dependence/proximity models (MRF [26], pBiL [29])	2

Table 1: Deployed features for document ranking.

While query independent features are retained in memory, the calculation of additional query dependent features after the first phase has ended requires access to the postings of the query terms. To facilitate this, we adapt WAND such that the matched postings of any document making the top K are retained in memory; any document expelled from the top K (due to another higher scoring document) has its retained postings discarded. As the postings contain frequencies, positions and field information, additional query dependent features can be efficiently calculated.

5.4 Learned Model Phase

We create learned models for the 33 document feature using 2 different learning to rank techniques namely:

Automatic Feature Selection (AFS) [27] obtains a weight for the linear combination of the most effective feature at each iteration to the features selected in the previous iteration(s). In our implementation, we use simulated annealing to find the combination weight for each feature that maximises NDCG. The weight of a newly selected feature is obtained without retraining the weights of those features already selected. The set of feature weights with the highest performing iteration on the validation data is chosen (in this manner, the validation data is used to determine the correct number of AFS iterations, as suggested by [20]).

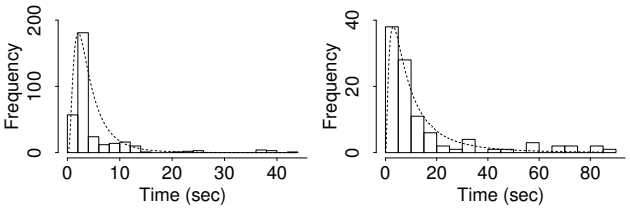
LambdaMART [37] deploys gradient boosted regression trees internally, while considering NDCG values within the training of the trees. We use the Jforests implementation [18].[4] A LambdaMART approach was the winning entry in the 2011 Yahoo! learning to rank challenge [9]. The model with the highest validation performance is chosen.

5.5 Measuring Efficiency

All efficiency experiments are made with a dual quad-core Intel Xeon 2.4GHz, with 32GB RAM, using a single monolithic index. Indeed, following Wang et al. [35], we did not use a distributed retrieval setting over multiple smaller indices coordinated by a broker (as might be deployed in a real-world system) as we did not wish to consider unrelated issues such as network latencies. Hence, while our query response times may be larger than would suffice for an interactive environment, this does not detract from the generality of the proposed selective pruning framework.

The mean response times (MRTs) of each of the retrieval phases for $K = 1000$ $F = 1$ are as follows: Top K Retrieval (12.9 s); Feature Extraction (18 ms); Learned Model Application (AFS: 0.71 ms; LambdaMART: 17.44 ms). Considering the fact that [35] directly uses the compression library of Terrier, our larger MRT for the first phase compared to [35] is expected as our index contains 4 additional frequencies per posting (for each field: title, body, anchor text & URL). Moreover, these response times justify our focus on the first retrieval phase to benefit efficiency, as the learned model application costs are negligible. Indeed, in applying the learned models of LambdaMART, we do not use the tree pruning method of Cambazoglu et al. [7], as the Feature Extraction and Learned Model Application phases have an insignificant duration compared to the initial Top K Retrieval phase.

[4] http://code.google.com/p/jforests/



(a) 2 terms (b) 4 terms

Figure 2: Response time distributions for 2 and 4 terms queries (boxes), and fitted log-normal distributions (dashed lines).

5.6 Effectiveness Prediction

Two forms of effectiveness prediction have been proposed in the literature [8]: *pre-retrieval* predictors are calculated solely on statistics from the query, without resorting to inverted index access [19]; in contrast, *post-retrieval* predictors have more evidence, such as the scores or contents of the retrieved documents [13]. In this work, as we require to estimate the effectiveness of a query before retrieval commences such that K and F parameters can be selected, we deploy only existing well-known pre-retrieval predictors, namely: AvICTF, AvIDF, $\gamma 1$ and $\gamma 2$ [19], Similarity Collection Query (SCQ), Normalised SCQ and Maximal SCQ [38]. These seven performance predictors are combined to estimate the effectiveness $\widehat{E}(q)$ of query q, by using gradient boosted regression trees (GBRT) [17] targeting NDCG, based on the 50 training and 50 validation effectiveness queries.

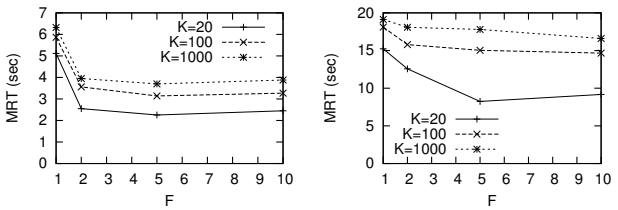
5.7 Efficiency Prediction

Query efficiency prediction [23] allows the response time of a query $R(q)$ to be accurately predicted before retrieval commences, denoted $\widehat{R}(q)$. Following Macdonald et al. [23], we calculate various statistics of each term at indexing time (e.g. maximum score, number of postings, number of posting with score less than 5% from maximum score), which are then combined into features using aggregated functions such as SUM and MAX for each constituent term of a query (more details can be found in [23]). We deploy GBRT [17] to learn the accurate response time predictions, based on the 500 efficiency training queries.

6. ON PRUNING AGGRESSIVENESS

In Section 4, we postulated two hypotheses concerning how query performance and efficiency predictions can be used to select which queries should be aggressively pruned by WAND. In this section, we aim to identify suitable definitions for SELECT($\bullet$) within the selective pruning framework of Algorithm 1. To do so, instead of considering all possible values $K > 1$ and $F \geq 1$, we use a set of discrete *candidate* settings that exhibit either good effectiveness or efficiency. In the following, we identify the candidate settings of K and F that the framework can select from in response to a query – this is in line with the methodology applied in previous experiments in identifying candidates within selective approaches (e.g. [2, 31]). Finally, we use the results of this analysis to instantiate definitions for SELECT$_{QPP}$($\bullet$) and SELECT$_{QEP}$($\bullet$).

To illustrate the potential for a selective approach that tackles inefficient queries (i.e. Hypothesis 2), Figure 2 shows the distribution of response times of WAND for queries of two different lengths. Other query lengths produce simi-



(a) 2 terms (b) 4 terms

Figure 3: Efficiency of different F and K for 2 and 4 terms queries.

lar, scaled distributions. From this, we can see that the response times for each query length approximately follow a log-normal distribution (shown as dashed lines), with a number of queries taking considerably longer than the most frequent response time. These are the queries that we target by the application of the selective pruning framework.

In the following, we analyse the impact of K and F on WAND efficiency (Section 6.1) and effectiveness (Section 6.2). In Section 6.3, we propose definitions for the $\text{SELECT}_{QPP}(\bullet)$ and $\text{SELECT}_{QEP}(\bullet)$ functions.

6.1 Impact on Efficiency

The K and F parameters depict the aggressiveness of pruning, and hence the resulting efficiency of WAND. Broder et al. [6] previously showed the efficiency impact of these parameters on a small corpus. In contrast, we examine these parameters on the 30 times larger ClueWeb09 corpus.

Reducing K: Reducing the K parameter of WAND for a given query causes the threshold τ to rise more quickly, with the effect of allowing pruning to occur more often. This benefits efficiency, as the number of scored postings is reduced and more skipping can be performed. Figure 3 shows the efficiency, measured by the mean response time (MRT) in seconds of the Top K Retrieval phase across the 981 real queries, for two different query lengths. The WAND strategy is used to retrieve the sample (with representative sample sizes of $K = \{20, 100, 1000\}$). From Figure 3, we can see that smaller K values have decreased mean response times.

Increasing F: Instead of reducing the number of retrieved documents to increase the threshold value, the threshold factor F can be used to directly increase the threshold. Indeed, increasing F requires that documents have to contain more query terms before they will be fully scored and admitted to the top K result set. Figure 3 also shows the benefit of different representative threshold factors ($F = \{1, 2, 5, 10\}$). Clearly, increasing F from 1 to 2 produces a marked efficiency benefit for shorter queries, however, this benefit is less marked for longer queries when K is large.

Recall that we are interested in identifying settings of K and F that lead to overall efficiency benefits. Figure 4 shows the reduction in MRT for various settings across various query lengths, compared to an initial, standard safe setting of $K = 1000$, $F = 1$. From the figure, we observe that increasing F from 1 to 2 has the most significant benefit on efficiency (up to 40% reduction in response time). However, $F = 5$ does not show as much benefit over $F = 2$. Thereafter, MRT improvements are only obtained for reducing K. Therefore the most efficient settings of WAND are $K = 20$ for some $F > 1$.

6.2 Impact on Effectiveness

Increasing F also results in an unsafe and possibly degraded effectiveness, as documents that should have been

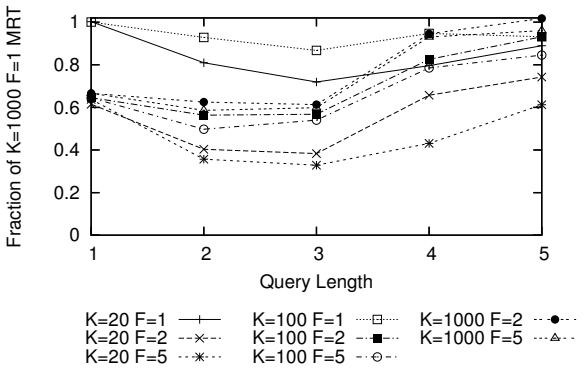


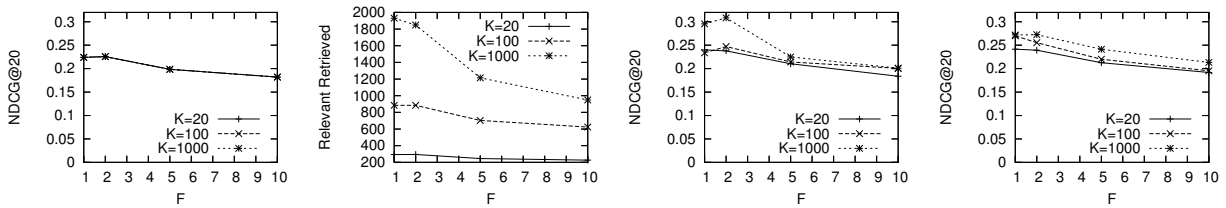
Figure 4: Efficiency improvements: Fraction of MRT exhibited by $K = 1000, F = 1$ achieved by various settings of K and F over 1000 queries.

retrieved in the top K may be omitted. Broder et al. [6] also showed how this impacted on the retrieval effectiveness, when directly measured on the resulting set of documents. However, in our scenario employing learning to rank, the output of K documents from WAND is used as the sample for re-ranking by the learned model. Indeed, varying K also has an impact on the resulting effectiveness of the model. For instance, a sample of size $K = 20$ used as input to a learned model will usually have lower effectiveness in terms of NDCG@20 than a sample of size $K = 100$, as the larger sample is likely to have higher recall, such that more relevant documents can be re-ranked within the top 20.

Figure 5 shows the effectiveness across the test set of the TREC Web track 2009-2011 queries for various K and F values. In particular, in Figure 5 (a), the NDCG@20 for three different sample sizes is plotted as F is varied. We firstly note that the $K = 20$, $K = 100$ and $K = 1000$ samples are super-imposed, as they identically rank documents up to rank 20. However, with respect to the threshold factor, effectiveness across all K values is reduced as F increases. In general, unsafe ranking does not impact on effectiveness to rank 20 for small values of F, which is in line with the findings of earlier experiments by Broder et al. [6]. In terms of recall, Figure 5 (b) shows that both reducing K and increasing F reduces the number of relevant documents identified.

In Figures 5 (c) & (d), the performances of the resulting learned models are shown for the AFS and LambdaMART learning to rank techniques, respectively. For different settings of K and F, we created new learned models for each learner. This is motivated in that the learned models are dependent on the size of the input sample of documents being ranked, since the probability of relevance within the sample increases as the sample size decreases [25]. For example, a learned model obtained on a small sample may apply a feature such as PageRank more aggressively than a learned model obtained from a larger sample. Note that retaining multiple learned models within the search engine is not a significant overhead, as they are lightweight to store and use at querying time, while their learning is conducted offline.

On analysing Figures 5 (c) & (d), we find that, in general, models from the two learners result in markedly increased effectiveness compared to the corresponding sample of the same K and F (Figure 5 (a)). The models are also affected by the decreasing recall of the sample, either as K decreases, or as F increases. In particular, analysing the impact of F, we find that effectiveness is generally decreased for $F > 2$. However, the effectiveness of a learned model obtained



(a) Sample (b) Sample Recall (c) AFS (d) LambdaMART

Figure 5: Effectiveness of the sample and learned models for different F and K.

for $F = 2$ is often comparable (or better) than for $F = 1$. This is caused by the indeterminate nature of unsafe retrieval: unsafeness does not necessarily imply degraded effectiveness by WAND. Indeed, an irrelevant document that is pruned from the sample for $F = 2$ may cause another relevant document to be retrieved that normally would not appear in the top K [24]. When evaluating the output of WAND directly using NDCG@20 (as in Figure 5 (a)), the impact of unsafe retrieval is not marked, as it rarely affects the top 20 documents. However, the effect of unsafe retrieval is, to some extent, magnified by the application of a learned model, which by re-ranking can promote documents from very low ranks in the sample.

Finally, we analyse the impact of K in Figures 5 (c) & (d). As expected, the learned models exhibit higher performance for $K = 1000$. Reducing this to $K = 100$ has some impact, but not as marked as for $K = 20$. This shows the importance of having sufficient relevant documents in the sample for an effective learned model, necessitating using a K as large as possible. Overall, we conclude that when evaluating for high precision (such as NDCG@20), adding a learned model to the output of WAND increases overall effectiveness. However, the learned models are also more sensitive to a large degree of unsafe retrieval (i.e. $F > 2$). Decreasing the number of K sample documents considered for re-ranking by the learned model also results in degraded effectiveness.

6.3 Definitions for SELECT

A search engine can take advantage of our observation that accurate search results can still be obtained when an unsafe ranking is used as input to the learned model (Section 6.2). However, while the use of degraded unsafe samples are desirable to ensure efficiency (Section 6.1), they should be avoided to maximise overall learned model effectiveness.

To use these observations for selecting appropriate WAND settings on a per-query basis, we firstly decide on the K and F values that are appropriate for candidate settings within the definitions of SELECT($\bullet$). Based on our analysis of Figures 3 & 4, we identify $K = 20, F = 2$ as a setting that produces the highest efficiency benefit with the least aggressive pruning. However, this setting can markedly degrade the effectiveness of learned models (see Figure 5 (c) and (d)). On the other hand, the $K = 1000, F = 1$ setting is safe, effective, but inefficient. Hence, $K = 1000, F = 1$ and $K = 20, F = 2$ form our candidate settings for WAND that can be selected by definitions of SELECT($\bullet$). Next, we propose definitions for SELECT($\bullet$) for our two hypotheses.

Selecting on Predicted Effectiveness

Our first hypothesis suggests that when a query is expected to be easier, where relevant documents are likely to be ranked highly in the sample, then the effectiveness of safe retrieval is likely to be sufficiently high, that unsafe retrieval can be applied to increase efficiency without perceivable effectiveness degradation. In contrast, if the query is expected to have few relevant documents, or that these are deeply ranked within the sample, then safe retrieval should be applied to preserve effectiveness. To support this, we examined the Spearman's ρ correlation between the effectiveness of learned models created from $K = 1000$, $F = 1$ and $K = 20$, $F = 2$. Across the 50 effectiveness test queries, we observed $\rho = 0.73$ for AFS and $\rho = 0.81$ for LambdaMART, suggesting that easier queries remain effective even when aggressively pruned. To estimate the difficulty of each query, we apply a combination of 7 pre-retrieval query performance predictors, as described in Section 5.6. Hence, for a newly arrived query q, we compare the estimated effectiveness $\widehat{E}(q)$ with a threshold ϵ to select an appropriate candidate setting of K and F:

$$\text{SELECT}_{QPP}(\text{PREDICT}(q)) = \begin{cases} \{20, 2\} & \text{if } \widehat{E}(q) > \epsilon \\ \{1000, 1\} & \text{otherwise} \end{cases}$$

If the predicted effectiveness $\widehat{E}(q)$ of query q is higher than threshold ϵ, the correct results are expected to be highly ranked in the sample, and hence effective retrieval should still be obtained with aggressive pruning ($K = 20, F = 2$). Otherwise, a safe but less efficient setting is selected for WAND. $0 < \epsilon < 1$ controls the number of queries aggressively pruned, where increasing ϵ reduces the number of applications of $K = 20, F = 2$.

Selecting on Predicted Efficiency

Hypothesis 2 suggests that when a query will not take a long time to complete, safe retrieval can be applied. In contrast, if the query terms have long postings lists and the query is difficult to prune, then the safe retrieval would take a significant amount of time, and more aggressive pruning should be applied. To decide which queries should be aggressively pruned, we recall that the response time distributions in Figure 2 follow log-normal distributions for each query length. With this in mind, we fit a log-normal distribution for each query length, centred around the geometric mean G_l for a given query length l. G_l represents a time above which we consider a query to be inefficient, obtained from the 500 efficiency training queries (see Section 5.1), while inefficient queries are identified using the query efficiency predictors, as described in Section 5.7. Hence, for a newly arrived query q, if the predicted response time $\widehat{R}(q)$ of the query is above the geometric mean, we select a more aggressive candidate K and F setting to increase the efficiency of the query:

$$\text{SELECT}_{QEP}(\text{PREDICT}(q)) = \begin{cases} \{20, 2\} & \text{if } \widehat{R}(q) > c \cdot G_l \\ \{1000, 1\} & \text{otherwise} \end{cases}$$

where $0 \leq c \leq 1$ is a parameter for adjusting the threshold above which a query will be aggressively pruned.

In the next section, by instantiating either $\text{SELECT}_{QPP}(\bullet)$ or $\text{SELECT}_{QEP}(\bullet)$ into the selective pruning framework of Algorithm 1, we evaluate the efficiency and effectiveness impacts of the two proposed definitions for SELECT($\bullet$).

7. EXPERIMENTS

In the following, we experiment to validate Hypotheses 1 and 2 in Sections 7.1 and 7.2, respectively, using the experimental setup defined in Section 5. In particular, we compare both approaches to the efficiency and effectiveness of four *Uniform* settings of K and F. These include the candidate settings of $K = 1000, F = 1$ and $K = 20, F = 2$, as well as two other settings ($K = 1000, F = 2$; $K = 100, F = 2$) from Section 6.2 that represent good tradeoffs of efficiency and effectiveness. Indeed, effectiveness is measured by NDCG@20 on 50 TREC Web track queries, while efficiency is measured by the mean and 90th percentile response times on the 481 efficiency queries (denoted MRT and 90%tl-RT, respectively). 90%tl-RT is an indicator of the response time suffered by users for the slowest 10% of queries. Following [35], we measure the significance of improvements by a learned model over the effectiveness of the unlearned safe sample (NDCG@20 0.224 from Figure 5 (a)), thereby using a common basis for all significance tests. In general, we consider a successful approach to be one with a significant improvement over the unlearned safe sample, while minimising MRT and 90%tl-RT.

7.1 Selecting on Predicted Effectiveness

Table 2 reports the efficiency and effectiveness of the uniform settings of K and F, as well as our selective approach based on query performance prediction. From the top part of the table, we observe that while the effectiveness of the uniform settings varies, an effective AFS learned model can significantly improve over the unlearned safe sample with $p < 0.01$ (denoted **), and an effective LambdaMART model can achieve a significant improvement over the sample at the 5% level ($p < 0.05$, denoted *). Hence, successful selective approaches should achieve such effectiveness improvements as a minimum – for both learners, this cannot be achieved with MRTs lower than 10.6 seconds.

To investigate the selective approach using query performance predictors – namely $\text{SELECT}_{QPP}(\bullet)$ – Table 2 reports the effectiveness, efficiency and percentage of queries aggresively pruned for various settings of the query performance predictor threshold ϵ. Indeed, recall that increasing ϵ decreases the number of predicted easy queries that will be aggressively pruned. The query performance prediction deployed in this work is a supervised combination of pre-retrieval predictors, and hence requires training data - as detailed in Section 5.6, we use the 50 training and 50 validation queries to learn to predict the NDCG@20 of the learned model. For this reason, the reported response times for $\text{SELECT}_{QPP}(\bullet)$ differ across the learned models in Table 2.

On examining the results, we note that for $\epsilon = 0.25$ for AFS and $\epsilon = 0.2$ for LambdaMART (denoted with dashed underlines in Table 2), significant NDCG@20 improvements over the safe sample can be obtained, with marked reductions in response time compared to the uniform setting of $K = 1000, F = 2$. In particular, for AFS, by aggressively pruning 77% of the 481 efficiency queries, a response time of 9.3s can be achieved, which is comparable to $K = 100$, $F = 2$, while still achieving higher NDCG@20 on the 50 effectiveness queries. For LambdaMART, a MRT of 9.0s is achieved – again, better than $K = 100, F = 2$ while still significantly improving over the effectiveness of the unlearned sample. In general, the results across AFS and LambdaMART are similar, with LambdaMART achieving the significantly improved effectiveness over the unlearned safe sample with slightly more aggressive pruning. In terms of 90th percentile response time, $\text{SELECT}_{QPP}(\bullet)$ is unable to improve upon the uniform $K = 100, F = 2$ setting, suggesting that while the aggressive pruning of predicted easy queries can enhance efficiency without significantly degrading effectiveness, it does not directly target the particularly inefficient queries.

Overall, we conclude that although selectively applying aggressive pruning on a per-query basis by using query performance predictors can improve response times while assuring effectiveness, it does so in a conservative manner. In contrast, in the next section, we show how directly targeting the inefficient queries is a more suitable approach for selective pruning.

7.2 Selecting on Predicted Efficiency

We now report the experimental results for $\text{SELECT}_{QEP}(\bullet)$. The deployed query efficiency predictors and the estimation of G_l require training data. These are trained on the 500 efficiency training queries, and efficiency results are reported on the separate testing set of 481 queries.

Table 3 reports both the efficiency and effectiveness results of our selective framework based on query efficiency prediction for estimating response times. To examine if the accuracy of the efficiency predictors impact upon the selective pruning framework, we report results when the actual response times are assumed to be known, i.e., using a perfect response time predictor $R(q)$, in addition to estimated response time $\widehat{R}(q)$.[5] We vary the c parameter of SELECT_{QEP} to measure its impact on efficiency and effectiveness.

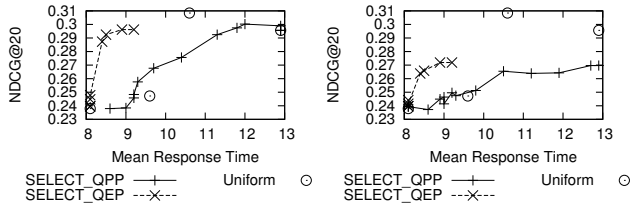
On examining Table 3, we note that varying the c parameter indeed affects efficiency and effectiveness, with the lowest c value providing the best response times but not attaining the significant improvements in effectiveness over the unlearned safe sample. In particular, we observe that with $c = 1$ (dashed underline in Table 3), very low response times can be achieved without aggressively pruning all queries, while still retaining high effectiveness by the learned models. For instance, for the AFS learner with the predicted response time, a MRT of 8.4s is achieved, which is close to the lower bound of 8.1s exhibited by the uniform aggressive pruning ($K = 20, F = 2$), while only aggressively pruning 35% of queries. Compared to the uniform safe setting ($K = 1000, F = 1$), this represents a 36% improvement in MRT. Moreover, we note that the 90th percentile response time drops 50% compared to the uniform safe setting of $K = 1000, F = 1$ and 24% compared to $K = 100, F = 2$ (19.3s vs. 38.3s and 24.9s, respectively). This attests that selecting K and F for each query based on predicted response time indeed targets the most inefficient queries.

Meanwhile the NDCG@20 of the AFS learned model for $c = 1$ still significantly improves over the unlearned safe sample at $p < 0.01$ - indeed, this is 21% higher than for $K = 20, F = 2$ (0.287 vs. 0.238), and 16% higher than the good uniform tradeoff setting of $K = 100, F = 2$ (viz. 0.247). For LambdaMART, similar results are observed, with the $c = 1$ setting offering a good tradeoff of effectiveness and efficiency. We note one exception for LambdaMART, where $c = 0.5$ still offers a significant improvement over the unlearned safe sample performance, but by a smaller absolute margin (9%). Finally, comparing the results using actual and predicted response times ($R(q)$ vs. $\widehat{R}(q)$), we note that using $R(q)$ only results in small improvements in

[5]Unlike in Table 2, the efficiency results of Table 3 are not dependent on the effectiveness of the learner on the training queries, thus the efficiency results are identical for both learners.

	AFS					LambdaMART				
	MRT	90%tl-RT	(%agg.)	NDCG@20	(%agg.)	MRT	90%tl-RT	(%agg.)	NDCG@20	(%agg.)
Uniform: same setting of K and F for all queries										
$K = 1000$ $F = 1$	12.9	38.3	-	0.296**		12.9	38.3	-	0.272*	-
$K = 1000$ $F = 2$	10.6	29.1	-	0.308**		10.6	29.1	-	0.273*	-
$K = 100$ $F = 2$	9.6	24.9	-	0.247*		9.6	24.9	-	0.256	-
$K = 20$ $F = 2$	8.1	17.8	-	0.238*		8.1	17.8	-	0.239	-
SELECT$_{QPP}(\widehat{E}(q))$: selective pruning framework applied using query performance predictors										
$\epsilon = 0.05$	8.6	25.2	(97%)	0.238*	(100%)	8.1	17.8	(97%)	0.239	(100%)
$\epsilon = 0.1$	9.0	26.2	(93%)	0.238	(94%)	8.6	25.2	(94%)	0.237	(96%)
$\epsilon = 0.15$	9.2	26.4	(89%)	0.246*	(91%)	8.9	26.2	(90%)	0.245	(94%)
$\epsilon = 0.2$	9.2	26.4	(85%)	0.248*	(83%)	9.0	26.2	(82%)	0.246*	(87%)
$\epsilon = 0.25$	9.3	26.4	(77%)	0.258**	(74%)	9.0	26.2	(75%)	0.241	(81%)
$\epsilon = 0.3$	9.7	27.0	(65%)	0.268*	(64%)	9.2	26.4	(69%)	0.250*	(70%)
$\epsilon = 0.35$	10.4	32.4	(48%)	0.276**	(51%)	9.3	26.4	(56%)	0.248	(60%)
$\epsilon = 0.4$	11.3	34.2	(33%)	0.293**	(38%)	9.8	26.4	(45%)	0.251*	(51%)

Table 2: Efficiency and effectiveness results of the selective pruning framework using query performance predictors. Significant improvements in effectiveness compared to the uniform application of the $K = 1000$ $F = 1$ sample (NDCG@20 0.224), as measured by the paired-t test, are denoted by * ($p < 0.05$) and ** ($p < 0.01$). The percentage of queries that experience aggressive pruning are shown in parentheses, denoted (%agg).



(a) AFS (b) LambdaMART

Figure 6: Comparison of efficiency and effectiveness for Hypotheses 1& 2 selective pruning approaches.

MRT, 90%tl-RT and NDCG, suggesting that our predicted response times are sufficiently accurate.

Overall, with respect to our second hypothesis, we find that the proposed approach for selective pruning based on estimated response times attains impressive efficiency benefits while still attaining high effectiveness, thereby validating Hypothesis 2. In particular, the results in this section suggest that queries that are predicted to be particularly inefficient typically represent those that can be more aggressively pruned, thereby ensuring efficiency, without damaging overall effectiveness. To aid in the comparison of our two selective approaches based on Hypotheses 1 & 2, Figure 6 shows the efficiency and effectiveness of each approach, where points nearest to the top-left are both efficient and effective. Indeed, we can see that the approach of Hypothesis 2 is markedly better for ensuring efficient yet effective retrieval on a per-query basis. In contrast, the approach of Hypothesis 1 is only slightly markedly more efficient and effective than the various used uniform settings. We posit that the success of selective pruning by query efficiency prediction can be explained in that the queries that take longer have longer posting lists (posting list length has a marked impact on efficiency [23]). Such queries may have more relevant documents, which are more likely to be found in the top-ranked sample, in contrast to more efficient queries.

8. CONCLUSIONS

This work proposed a selective pruning framework for ensuring efficient yet effective retrieval, by appropriately setting the pruning parameters of WAND on a per-query basis, before re-ranking the results using a learned model. Two alternative approaches are proposed within our selective pruning framework for selecting queries to prune, by using query performance predictors to identify easier queries, and by using query efficiency predictors to identify inefficient queries. Our experimental results showed that using the proposed selective pruning framework in conjunction with query efficiency predictors to select the queries to aggressively prune gave the best overall results. In particular, it could improve mean response time by 36% and the response time experienced by the slowest 10% of queries by 50%, while still maintaining significantly high effectiveness.

Our proposed selective pruning framework can have further applications. In particular, when the search engine is receiving a high volume of queries, pruning can be made more aggressive to ensure that a maximum allowed response time is adhered to. We will address this in future work. Moreover, we would like to investigate loss functions that permit efficiency and effectiveness to be considered within a learned approach for obtaining appropriate K and F settings on a per-query basis. A key practical challenge in this respect is the availability of large query sets with corresponding relevance assessments, to permit efficiency and effectiveness to be measured for the same queries.

9. REFERENCES

[1] G. Amati, E. Ambrosi, M. Bianchi, C. Gaibisso, G. Gambosi. FUB, IASI-CNR and Univ. of Tor Vergata at TREC 2007 Blog track. In *Proc. of TREC*, 2007.

[2] G. Amati, C. Carpineto, G. Romano. Query Difficulty, Robustness, and Selective Application of Query Expansion. In *Proc of ECIR*, 2004.

[3] V. N. Anh, A. Moffat. Pruned query evaluation using pre-computed impact scores. In *Proc. of SIGIR*, 2006.

[4] M. Bendersky, W. B. Croft, Y. Diao. Quality-biased ranking of web documents. In *Proc. of WSDM*, 2011.

[5] R. Blanco. *Index compression for information retrieval systems*. PhD thesis, Univ. of A Coruna, 2008.

[6] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc of CIKM*, 2003.

[7] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proc. of WSDM*, 2010.

	AFS					LambdaMART				
	MRT	90%tl-RT	(%agg.)	NDCG@20	(%agg.)	MRT	90%tl-RT	(%agg.)	NDCG@20	(%agg.)
Uniform: same setting of K and F for all queries										
$K = 1000$ $F = 1$	12.9	38.3	-	0.296**	-	12.9	38.3	-	0.272*	-
$K = 1000$ $F = 2$	10.6	29.1	-	0.308**	-	10.6	29.1	-	0.273*	-
$K = 100$ $F = 2$	9.6	24.9	-	0.247*	-	9.6	24.9	-	0.256	-
$K = 20$ $F = 2$	8.1	17.8	-	0.238*	-	8.1	17.8	-	0.239	-
SELECT$_{QEP}(R(q))$: selective pruning framework applied using actual response times										
$c = 0.1$	8.1	17.8	(99%)	0.240*	(98%)	8.1	17.8	(99%)	0.240	(98%)
$c = 0.25$	8.1	17.8	(98%)	0.240*	(98%)	8.1	17.8	(98%)	0.240	(98%)
$c = 0.5$	8.1	17.8	(77%)	0.245**	(89%)	8.1	17.8	(77%)	0.244*	(89%)
$c = 0.75$	8.1	19.0	(51%)	0.248*	(68%)	8.1	19.0	(51%)	0.242	(68%)
$c = 1$	8.2	19.0	(36%)	0.290**	(32%)	8.2	19.0	(36%)	0.268*	(32%)
$c = 1.25$	8.4	19.3	(31%)	0.296**	(23%)	8.4	19.3	(31%)	0.272*	(23%)
$c = 1.5$	8.7	19.3	(25%)	0.296**	(23%)	8.7	19.3	(25%)	0.272*	(23%)
$c = 2$	9.1	21.9	(20%)	0.296**	(17%)	9.1	21.9	(20%)	0.274*	(17%)
SELECT$_{QEP}(\widehat{R}(q))$: selective pruning framework applied using predicted response times										
$c = 0.1$	8.1	17.8	(100%)	0.240*	(98%)	8.1	17.8	(100%)	0.240	(98%)
$c = 0.25$	8.1	17.8	(95%)	0.240*	(98%)	8.1	17.8	(95%)	0.240	(98%)
$c = 0.5$	8.1	18.9	(78%)	0.245**	(89%)	8.1	18.9	(78%)	0.244*	(89%)
$c = 0.75$	8.1	19.0	(53%)	0.248*	(70%)	8.1	19.0	(53%)	0.242	(70%)
$c = 1$	8.4	19.3	(35%)	0.287**	(34%)	8.4	19.3	(35%)	0.264*	(34%)
$c = 1.25$	8.5	19.3	(31%)	0.293**	(28%)	8.5	19.3	(31%)	0.266*	(28%)
$c = 1.5$	8.9	19.3	(25%)	0.296**	(21%)	8.9	19.3	(25%)	0.272*	(21%)
$c = 2$	9.2	21.0	(19%)	0.296**	(21%)	9.2	21.0	(19%)	0.272*	(21%)

Table 3: Results of the selective pruning framework using query efficiency predictors. Notation as in Table 2.

[8] D. Carmel and E. Yom-Tov. Estimating the Query Difficulty for Information Retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1), 2010.

[9] O. Chapelle, Y. Chang. Yahoo! learning to rank challenge overview. *J. Machine Learning: Workshop and Conference Proceedings*, 14:1–24, 2011.

[10] C. L. A. Clarke, N. Craswell, I. Soboroff, G. V. Cormack. Overview of the TREC 2010 Web Track. In *Proc. of TREC*, 2010.

[11] G. V. Cormack, M. D. Smucker, C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large Web datasets. *Inf. Retr.*, 14(5):441–465, 2011.

[12] N. Craswell, R. Jones, G. Dupret, E.Viegas, editors. Proc. of the Web Search Click Data Workshop at WSDM 2009.

[13] S. Cronen-Townsend, Y. Zhou, W.B. Croft. A Framework for Selective Query Expansion. In *Proc. of CIKM*, 2004.

[14] J. Dean. Challenges in building large-scale information retrieval systems: invited talk. In *Proc. of WSDM*, 2009.

[15] R. Fagin, A. Lotem, M. Naor. Optimal aggregation algorithms for middleware. *J. Computer and System Sciences*, 66(4):614–656, 2003.

[16] M. Fontoura, V. Josifovski, J. Liu, S. Venkatesan, X. Zhu, J. Zien. Evaluation strategies for top-k queries over memory-resident inverted indexes. *Proc. VLDB Endowment*, 4(12):1213–1224, 2011.

[17] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[18] Y. Ganjisaffar, R. Caruana, C. Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proc. of SIGIR*, 2011.

[19] B. He, I. Ounis. Inferring query performance using pre-retrieval predictors. In *Proc. of SPIRE*, 2004.

[20] T.-Y. Liu. Learning to rank for IR. *Foundation and Trends in IR*, 3(3):225–331, 2009.

[21] C. Macdonald, I. Ounis, N. Tonellotto. Upper bound approximations for dynamic pruning. *ACM Trans. Inf. Sys.*, 29(4):1–28, 2011.

[22] C. Macdonald, V. Plachouras, B. He, C. Lioma, I. Ounis. Univ. of Glasgow at WebCLEF 2005: Experiments in per-field normlisation and language specific stemming. In *Proc. of CLEF*, 2005.

[23] C. Macdonald, N. Tonellotto, I. Ounis. Learning to predict response times for online query scheduling. In *Proc. of SIGIR*, 2012.

[24] C. Macdonald, N. Tonellotto, I. Ounis. Effect of dynamic pruning safety on learning to rank effectiveness. In *Proc. of SIGIR*, 2012.

[25] C. Macdonald, R. Santos, I. Ounis. The whens and hows of learning to rank. *Information Retrieval*, 2012.

[26] D. Metzler, W. B. Croft. A Markov random field model for term dependencies. In *Proc. of SIGIR*, 2005.

[27] D. Metzler. Automatic feature selection in the Markov random field model for information retrieval. In *Proc. of CIKM*, 2007.

[28] A. Moffat, J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Sys.*, 14(4):349–379, 1996.

[29] J. Peng, C. Macdonald, B. He, V. Plachouras, I. Ounis. Incorporating term dependency in the DFR framework. In *Proc. of SIGIR*, 2007.

[30] M. Persin. Document filtering for fast ranking. In *Proc. of SIGIR*, 1994.

[31] V. Plachouras, I. Ounis. Usefulness of hyperlink structure for query-biased topic distillation. In *Proc. of SIGIR*, 2004.

[32] R. L. T. Santos, C. Macdonald, I. Ounis. Intent-aware search result diversification. In *Proc. of SIGIR*, 2011.

[33] E. Shurman, J. Brutlag. Performance related changes and their user impacts. In *Velocity: Web Performance and Operations Conf.*, 2009.

[34] H. Turtle, J. Flood. Query evaluation: strategies and optimizations. *Inf. Proc. Mngmnt*, 31(6):831–850, 1995.

[35] L. Wang, J. Lin, D. Metzler. Learning to efficiently rank. In *Proc. of SIGIR*, 2010.

[36] L. Wang, J. Lin, D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proc. of SIGIR*, 2011.

[37] Q. Wu, C. J. C. Burges, K. M. Svore, J. Gao. Ranking, boosting, and model adaptation. Technical Report MSR-TR-2008-109, Microsoft, 2008.

[38] Y. Zhao, F. Scholer, Y. Tsegay. Effective pre-retrieval query performance prediction. In *Proc. of ECIR*, 2008.