Using an Inverted Index Synopsis for Query Latency and Performance Prediction

NICOLA TONELLOTTO, University of Pisa
CRAIG MACDONALD, University of Glasgow

Predicting the query latency by a search engine has important benefits, for instance, by allowing the search engine to adjust its configuration to address long-running queries without unnecessarily sacrificing its effectiveness. However, for the dynamic pruning techniques that underlie many search engines, achieving accurate predictions of query latencies is difficult. We propose the use of index synopses – which are stochastic samples of the full index – for attaining accurate timings. Indeed, we experiment using the TREC ClueWeb09 collection, and a large set of user queries, to find that using small index synopses it is possible to very accurately estimate properties of the larger index, including sizes of posting list unions and intersections. Thereafter, we demonstrate that index synopses facilitate two use cases: firstly, for query efficiency prediction, we show that predicting the query latencies on the full index and classifying long-running queries can be accurately achieved using index synopses; Secondly, for query performance prediction, we show that the effectiveness of queries can be estimated more accurately using a synopsis index post-retrieval predictor than a pre-retrieval predictor. Overall, our experiments demonstrate the value of such a stochastic sample of a larger index at predicting the properties of the larger index.

Additional Key Words and Phrases: inverted index synopsis, query efficiency prediction, query performance prediction, dynamic pruning

ACM Reference Format:
Nicola Tonellotto and Craig Macdonald. 2020. Using an Inverted Index Synopsis for Query Latency and Performance Prediction. *ACM Transactions on Information Systems* , (June 2020), 32 pages.

1 INTRODUCTION

Information retrieval (IR) systems need to respond quickly to users' queries to ensure the users' satisfaction, and avoid large infrastructure costs. Query processing optimizations such as dynamic pruning techniques [31] can be deployed to ensure that the search engine can answer queries with low response times. Such techniques work by processing posting lists in disjunctive (OR) mode and by skipping the scoring of postings that cannot be retrieved in the top K documents for that querying – this is achieved by processing such posting lists in conjunctive (AND) mode.

Different query processing optimizations might be applied for different queries. For example, different query rewritings can be applied to the same query, with different cutoffs, depending on the specific terms appearing in the query [24]. Moreover, there are different optimizations of the search process and relative infrastructure that can be tuned, depending on the queries. For example, dynamic pruning strategies can be made more aggressive when the query load (i.e. the arrival rate of queries to be processed) increases [3], and the speed of the underlying processors in search nodes can be varied depending on the query load to save energy [10].

The deployment of such per-query optimization techniques relies on the availability of accurate estimates of the processing time of queries in the different operational scenarios of the search engine. Query efficiency prediction (QEP) [23] is an approach to predict the processing time of queries in search systems before the query is actually

*Authors preprint - accepted to ACM Transactions on Information Systems - Final version available at https://doi.org/10.1145/3389795

Authors' addresses: Nicola Tonellotto, University of Pisa, Department of Information Engineering, Via G. Caruso 16, 56122, Pisa, Italy, nicola.tonellotto@unipi.it; Craig Macdonald, University of Glasgow, G12 8QQ, Glasgow, UK, craig.macdonald@glasgow.ac.uk.

processed. QEP can be used to optimize the execution of queries in a number of different scenarios and with different goals. For example, QEP has been used to apply selectively dynamic pruning strategies [30] and to identify long-running queries before they are processed, and to process them using multiple processors in parallel [19]. In distributed search infrastructures, QEP has been used to improve the throughput of query processing servers [23], as well as to reduce the energy consumption of query processing servers [10].

Predicting accurately the execution time of a query before it is processed by a dynamic pruning technique is not straightforward. Indeed for a given query processing strategy, there are a number of factors that can affect the execution time of a query: these include simple factors such as the number of query terms composing the query and the length of the posting lists of those terms [23]; or more complex factors such as the differences in the query terms' importances, the number of documents in which they co-occur [34], and the portion of the posting lists that the dynamic pruning technique processes in conjunctive mode. Indeed, the execution time of a query depends on the postings that are actually accessed, decompressed and processed in the query terms' posting lists [23]. Very important query terms, i.e., terms appearing in few documents, have short posting lists, and typically most of their postings are processed. On the contrary, the postings of common terms, with very long posting lists, may not be fully processed by dynamic pruning technique. Nevertheless, most dynamic pruning strategies always process the postings of documents in which all query terms appear, and hence both the sizes of the intersection and unions of the postings lists will impact upon the processing time of the query.

Different techniques to predict the processing time of a query have been proposed, which differ in their phase of operation: *prior* to the start of processing (static QEP) [23], or *delayed* after the start of processing (dynamic QEP) [20]. In static QEP, the posting lists of an inverted index are pre-processed at indexing time, and a set of static term features are extracted and stored in the index. Given a query, the features of the query terms are combined using aggregation operators (e.g. sum, max, min). Then, a set of training queries is used to train a machine-learned model able to predict the execution times of unseen queries, given the aggregations of the static term features. On the other hand, in dynamic QEP, after a few milliseconds of query execution, e.g., 10 msec, the query features are computed, and used within a machine-learned model to predict the processing times. Indeed, delayed prediction can observe the correlation of query terms during retrieval, and hence can make more accurate response time predictions compared to a priori prediction. In this work, we take a different viewpoint, and make predictions based on a smaller *synopsis index*, i.e., an inverted index for a small randomly sampled percentage of the whole index. Predictions made using a synopsis index are shown to be accurate and inexpensive.

This work contributes a first study into index synopses, which we show are very useful at predicting the properties of the large (full-size) index, such as the sizes of unions or intersections of posting lists. In particular, we use index synopses for two use cases: firstly, for QEP, predicting the response time of the search engine for a query on the larger index; and secondly, predicting the *effectiveness* of the search engine for a query on the larger index (also known as query performance prediction QPP). While our experiments address a single retrieval index scenario, our techniques could easily be scaled to address distributed and replicated search settings.

In doing so, our concrete contributions are as follows:

- We provide a novel view on the processing of a query over a set of posting lists when dynamic pruning strategies are employed. In particular, as posting lists are traversed, we show that they are processed with a different mix of AND/OR modes for different consecutive docids.

- We propose a compact data structure aiming at reproducing the dynamic pruning behavior of the MaxScore, WAND and BMW strategies on a full inverted index. This data structure, that we call an (inverted) index synopsis, allows to estimate properties of the larger index, including sizes of posting list unions and intersections, as well as the response times.
- We show that an index synopsis can be used to obtain accurate query efficiency predictions for MaxScore, WAND and BMW dynamic pruning strategies, to estimate the processing times of queries on the full index. Our experiments, conducted using 50,000 unique queries from the TREC05 Web track on the TREC ClueWeb09B corpus, demonstrate that accurate query efficiency predictions for dynamic pruning strategies can be made using a very small synopsis index, containing no more than 1% of the documents in the original index.
- We propose that an index synopsis can be used to obtain post-retrieval query performance predictors. Indeed, our experiments, conducted using 200 TREC Web track queries on the TREC ClueWeb09B corpus, demonstrate that post-retrieval query performance predictors calculated on an index synopsis of only 1% are highly correlated with the same predictors evaluated on the full index, can outperform pre-retrieval query performance predictors.

The outline of the remainder of this paper is as follows: Section 2 provides related work on dynamic pruning and on other succinct index representations; Section 3 provides an overview of dynamic pruning; Section 4 presents our analysis on how dynamic pruning behavior changes at query processing time for different strategies. We describe our approach for query efficiency prediction on index synopses in Section 5; We detail the addressed research questions in Section 6 followed by experimental setup in Section 7; Thereafter follows detailed experiments addressing: the overheads of synopsis indices (Section 8); how well they capture properties of a larger index (Section 9); as well as their usefulness for query efficiency prediction (Section 10) and query performance prediction (Section 11). Finally, we summarize our findings and provide directions for future work in Section 12.

2 RELATED WORK

In the following, we review related work on predicting the response time of a search engine (Section 2.1), as well as positioning our work with respect to previous work on succinct index representations (Section 2.2).

2.1 Query Efficiency Prediction for Dynamic Pruning

As detailed in Section 1, query efficiency prediction – predicting how long a query will take to execute – can have a number of benefits, allowing to vary the configuration of the search engine for long-running queries [19, 24, 30]. For an exhaustive query processing technique, query efficiency prediction is trivial, as scoring must take place for all of the postings for every query term – as the length of the posting lists are known a priori, the time can be easily predicted [23]. However, for dynamic pruning techniques such as MaxScore, WAND, and BMW, there is more difficulty, as the proportion of each term's posting lists processed as essential and non-essential will vary, for instance according to the *pruning difficulty* of a given query. Macdonald et al. [23] treated query efficiency prediction as a supervised learning task, using pre-computed term-level features such as the length of the posting lists, and the variance of scored postings for each term. They reported Pearson correlations exceeding 0.9 on 10,000 queries. Jeon et al. [19] later adapted this approach for accurately classifying long-running (tail) queries (i.e., as a classification task rather than a regression task), and demonstrated its benefits on the Bing search engine infrastructure. Other more recent work [24] considered the challenge of query efficiency prediction for rewritten queries that include complex query operators such as `#1()` (proximity) or `#syn()` (synonyms).

Wu & Fang [34] developed an analytical model of query processing efficiency, noting that a key factor in their model was the number of documents containing pairs of query terms. Indeed, they noted that the threshold θ will rise more quickly if two query terms are correlated, i.e., they occur together. As it was considered infeasible to store term co-occurrence statistics, the authors assumed an approximate mixture between independence and dependence. As a consequence, for every term they propose to store the empirical mean and variance of the scores appearing in the term's posting list. The estimation of the number of postings processed in conjunctive mode lies at the core of the analytical performance modeling for top K query processing. In the model, it is assumed that, for two query terms t_1 and t_2 with posting lists of length N_1 and $N_2 \geq N_1$ respectively, the number of documents containing both terms $A(t_1, t_2)$ is computed with the following *analytical approximation*:

$$A(t_1, t_2) = \frac{N_1}{N} \times \left(\frac{N_2}{N}\right)^{\delta} \times N, \tag{1}$$

where N is the number of documents in the whole collection and δ is a parameter used to control how related the two terms are. The authors suggest that $\delta = 0.5$ is a good general value. Later, in Section 9, we show that using $\delta = 0.5$, the analytical model estimations of intersection sizes have excessively high variances.

To address the challenge of unreliable query efficiency predictions, Kim et al. [20] extended their earlier work [19] to make query efficiency predictions after a short period of query processing has elapsed. In doing so, their *delayed predictors* are able to determine how well a query is progressing, and used that to better estimate the query's completion time. Indeed, the increased accuracy of a delayed prediction is intuitive, as the correlation between the query terms can be better approximated after a portion of the index has been processed.

Even if delayed prediction is more accurate than predictions based on pre-computed features or an analytical model, it requires a supervised machine-learned model to perform predictions. This model must be periodically re-trained as new queries arrive. Moreover, the dynamic features are naturally biased towards the first portion of the index used to calculate them. Indeed, with various index orderings possible (random, URL ordering, PageRank ordering etc.), it is plausible that the first portion of the index does not reflect well the term distributions in the rest of the index.

Instead of using a first portion of the index to calculate, in this work we propose to use a probabilistic random sample of the index, called a *synopsis*. Furthermore, as our index is a stochastic sample of the full index, we contrast it with a statically pruned index, which is designed for retrieval by only including documents predicted to be relevant to more queries. Our experiments demonstrate that accurate predictions of response times can be attained using a synopsis index, at very small cost. Below, we position our proposal of synopsis indices viz. other succinct index representations in the literature.

2.2 Compact Index Representations

Our proposal of a synopsis index bears only surface similarities to other approaches from the literature that used compact (smaller) index representations. Firstly, we contrast our approach with *lossless* compressions of inverted index structures, such as Elias-Gamma, PForDelta, or Elias-Fano, that encode streams of integers using minimal numbers of bit – these are very efficient and widely used [31][1]. Indeed, a synopsis index is a *lossy* compression of the original index, where some information about is lost about which documents match which terms.

Static pruning techniques [6, 7] have been proposed to create smaller index representations, as a form of lossy compression of the inverted index. In doing so, such indices are designed to be much smaller and hence faster to retrieve

[1]Some of the compression advantages occur because it is the difference between two adjacent docids that are recorded, known as *dgaps* [31]. As dgaps are smaller than the original docids, they can be recorded in smaller numbers of bits

from than the full index, while still retaining sufficient potentially relevant documents in order to attain high user satisfaction. In practice, there is typically a loss of effectiveness when using a statically pruned index.

Static pruning techniques work by scoring the entire index using the weighting model, to identify documents or postings that can are unlikely to make the top K retrieved set. These documents/postings are then removed from the index. For instance, in the *document-centric* pruning technique of Büttcher and Clarke [6], for each given document, only the most important terms in each document will be kept. A synopsis index differs from document-centric pruning in that many (even most) documents will cease to exist in the index.

We also note surface similarities with a top-docs index [5, 29] - this is a set of the highest scoring documents for each term that have been stored separately from the large inverted index. These allow for efficient retrieval for a given query by taking the union of term-specific candidates lists.

Both static pruning and top-docs approaches differ from an index synopsis in that they actually put more effort into creation of the compact index, in order to maintain its effectiveness. Instead, our synopsis index is a simpler small random sample of the full index, which we show exhibits desirable properties of the larger index, and permits for both query efficiency prediction and query performance prediction for dynamic pruning.

Random samples of indexes are used also in federated search systems, where heterogeneous search engines (each with its own index) are managed as a single search service. In this scenario, each search engine is represented by a small random sample of its own document collection. These samples are merged together into a centralized sample index and used to select, for a given query, which underlying search engine should process it [21]. Our synopsis index has a completely different purpose, as it aims at providing quick estimates of posting list unions and intersections to approximate the results of identical computations performed on the full index.

Random sampling to estimate the cardinality size of unions and intersections of sets has been used for query optimization in database systems [15]. However, as we discuss in Sec. 4, dynamic pruning strategies require the union and/or intersection of different portions of the posting lists of a given query during query processing. In databases, cardinality estimates require to computed just the intersection and union sizes of given lists of objects. In a similar vein, in IR, Bharat and Broder [2] sampled random pages from the index of a search engine and checked whether they existed in other search engine indices, with a view to measuring the size and overlap of search engine indices. However, to the best of our knowledge, no work in IR has addressed the use of random samples for the response time prediction of dynamic pruning techniques, or for query performance prediction.

The sampling of larger indices has also seen work by those concerned with the evaluation of IR systems. For instance, Sanderson et al. [27] examined how effectiveness of systems differed across various sub-sets of TREC test collections – in this case, partitioned by document provider (e.g., newspaper). In contrast, Voorhees et al. [33] “randomly” partitioned the document set in a test collection, to evaluate the performance of systems on each partition, with a view to more robustly identifying significant differences between systems. However, Voorhees et al. noted that the number of relevant documents is the driving factor in the “statistical precision of a retrieval measurement”, and hence balanced the number of relevant documents among their partitions. For the same reason, we argue that random sampling of a larger index is unlikely to bring effective results, due to the comparative rarity of relevant documents. Instead, our work is primarily concerned with prediction, particularly of efficiency, but also of effectiveness. We do not consider further the random sampling of indices for the purposes of effective retrieval.

In the next sections, we provide necessary background on dynamic pruning (Section 3), we illustrate how unions and intersections of (portions of) posting lists are performed at query processing time (Section 4), and then define our synopsis index (Section 5), and describe how it is used for query efficiency and performance predictions.

3 DYNAMIC PRUNING

Ranked retrieval is concerned with retrieving the K highest scored documents, where the score is a function of the query-document pair. In doing so, a query can be processed in either conjunctive (AND) or disjunctive (OR) mode, retrieving the documents that contain respectively all the terms or at least one of them.

Dynamic pruning improves the efficiency of query processing in disjunctive ranked retrieval, by modifying the behavior of the retrieval algorithms to skip over documents that cannot reach a sufficient score to enter the final top K results. Thus, the final result is the same as exhaustive processing, but obtained significantly faster. Such techniques include MaxScore [32], WAND [4], and BMW [13].

Dynamic pruning strategies leverage *term upper bounds* and top K *thresholds*. A term upper bound (also known as a *max score*) σ_t is the score of the highest scoring document among those in the posting list of a given term t. During processing, the top K scores computed so far, and their corresponding docids, are organized in a min-heap data structure, and the smallest value of these scores is called the threshold θ. When a new document is scored, it can only have a chance to enter the final top K results if its score (or an upper bound on the score) exceeds the current threshold θ. During the processing of a query, the threshold θ starts at 0 and never decreases.

Given a document collection, the term upper bounds can be computed at indexing time. At query processing time, given a query q and a document d, we can compute the *document upper bound* $\sigma_d(q)$ by considering the query terms appearing in the document, and summing up the corresponding term upper bounds:

$$\sigma_d(q) = \sum_{t \in q \cap d} \sigma_t . \tag{2}$$

During query processing, the final query-document score is computed sequentially with the query terms. When a query term t is processed, the document upper bound in Eq. (2) is updated, by substituting the term upper bound σ_t with the actual term score. In doing so, the document upper bound decreases until all query terms have been processed, and the document upper bounds corresponds to the actual document score.

Note that the score computation of a given document can be early terminated as soon as its document upper bound decreases to a value lower than the current threshold θ. When the document does not include all query terms, i.e., $q \cap d \neq q$, it is possible to completely skip over the processing of the document. In fact, in that case, the document upper bound computed with Eq. (2) can be smaller than the threshold even before any actual term score is computed.

The main difference among MaxScore, WAND, and BMW strategies is the manner that documents to be fully or partially processed are identified. For more details on MaxScore, WAND, and BMW, we refer the reader to the recent review article on query processing [31].

In the next section we provide a novel view on the processing of a query over a set of posting lists when dynamic pruning strategies are employed. In particular, as posting lists are traversed, we will show that they are processed in with a different mix of AND/OR modes for different consecutive docids.

4 THRESHOLDS AND CRITICAL DOCIDS IN DYNAMIC PRUNING

The number of postings processed by dynamic pruning strategies strongly correlated to the query response times, as discussed in [23]. In the following, we characterize the dynamic pruning strategies in terms of intersections and unions of portions of posting lists, in such a way that, the number of processed postings depends on the posting lists processed in disjunctive and conjunctive mode.

Independently from their specific implementation details, all dynamic pruning strategies start processing all posting lists in disjunctive mode, and then they try to find the opportunity to avoid processing all postings of all posting lists.

The MaxScore dynamic pruning strategy separates the posting lists of each query term into two groups: essential lists, that must be processed in OR mode, and non-essential lists, that can be processed in AND mode. At the beginning, all posting lists are essential; the goal of MaxScore is to make most of the posting lists non-essential, and to do so as soon as possible. Any document appearing in an essential posting list has a chance to be returned in the final top K results, and hence those postings lists must be processed in OR mode. On the other hand, documents appearing in the non-essential posting lists alone will never accumulate enough score to be included in the current top K results, hence they must be processed in AND mode. As a consequence, any dynamic pruning strategy identifies the top K documents by processing exhaustively the essential lists, and, using the documents identified in these lists, skips over the other documents appearing in the non-essential lists only.

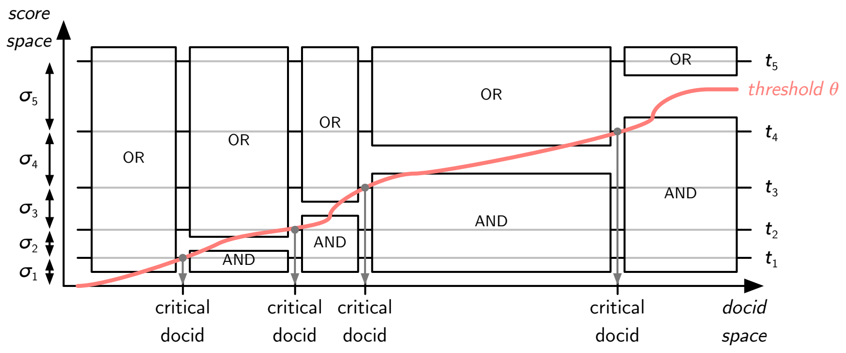


Fig. 1. An example of the MaxScore strategy with a query with 5 terms.

An example of the MaxScore dynamic pruning process is illustrated in Fig. 1, where a query with 5 terms is being processed. During query processing, the MaxScore strategy maintains the posting lists ordered by increasing term upper bounds on the *score space*. This ordering is specified by the query terms and their associated upper bounds, and does not change during query processing. Query processing proceeds in the *docid space*, in increasing docid order. Initially the value of the threshold θ is 0, but as documents are processed, the value increases. When it reaches the term upper bound of posting list of term t_1, it is guaranteed that no document containing only t_1 can ever be included in the final top K documents. Analogously, when the threshold reaches the sum of term upper bounds of terms t_1 and t_2, it is guaranteed that no document containing only t_1 and/or t_2 can ever be included in the final top K documents, and so on.

When the threshold values become equal to the sum of some term upper bounds, we call the docid being processed *critical docid*. Each time a critical docid is encountered, we have a split of the posting lists into essential and non-essential lists: essential lists are then processed in disjunctive mode and non-essential lists in conjunctive mode until a new critical docid, if any, is encountered.

The behavior of the MaxScore query processing changes when the threshold becomes greater than certain values, corresponding to the critical docids, which we call *critical values*, as in [34]. These critical values correspond to the partial sums of term upper bounds when the posting lists are sorted in increasing upper bound order.

The WAND dynamic pruning strategy adopts a different approach, leveraging the term upper bounds to identify portions of the docid space that have a chance to contain documents with a score greater than the current threshold.

Note that in WAND there is no distinction between essential and non-essential lists. The posting lists are not maintained sorted by increasing term upper bound, but are instead maintained ordered by increasing current docid. During processing, a *pivot term* is identified, by summing up its upper bound with all upper bounds of terms sorted before it, when the result is greater than the current threshold. The docid in the pivot term's posting list is the *pivot docid*: the posting lists of terms sorted before the pivot term posting list are skipped up to the pivot docid, hence processed in conjunctive mode, and the pivot docid is processed, together with the remaining posting lists, processed in disjunctive mode. Then, the strategy proceeds to the next document.

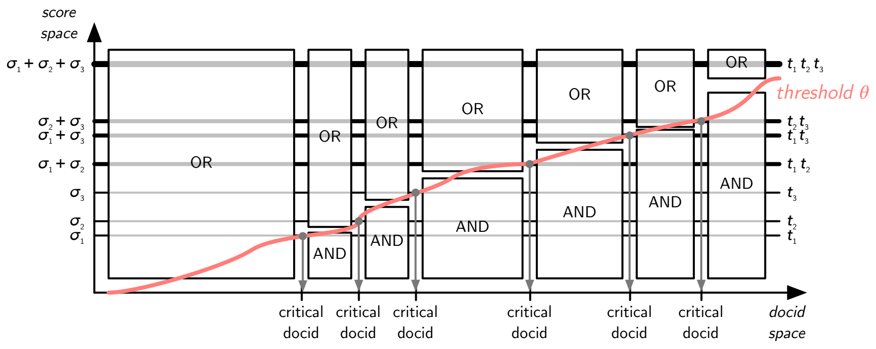


Fig. 2. An example of the WAND strategy with 3 terms.

Another way to consider the processing of posting lists carried out by the WAND strategy consists of considering all possible subsets of terms that can be processed together in conjunctive mode. An example of the WAND dynamic pruning process is illustrated in Fig. 2. When processing a query $q = \{t_1, t_2, t_3\}$, there are $2^3 - 1 = 7$ non-empty subsets of terms. Every subset of terms will have a subset upper bound given by the sum of the upper bounds of its constituent terms. Let these subsets of terms be sorted by increasing subset upper bound in the score space (y axis in Fig. 2). Initially the value of the threshold θ is 0, and all subsets are processed together in disjunctive mode, since any posting in any subset has a upper bound greater than the threshold. When the threshold increases up to the value of the first subset, containing the term t_1 only, a *critical docid* is encountered, with a corresponding *critical value*. From now on, no documents containing the term t_1 only will ever be returned in the final top K documents, hence the posting list of term t_1 will be processed in conjunctive mode with the remaining terms only, corresponding to the term subsets $\{t_1, t_2\}$, $\{t_1, t_3\}$ and $\{t_1, t_2, t_3\}$. When processing a query q with $|q|$ terms, the WAND query processing behavior changes when the threshold reaches the *critical values* corresponding to the $2^{|q|} - 1$ subset upper bounds of all the non-empty subsets composed by the terms of the original query q except the full query. As in the MaxScore strategy, the query terms and their associated upper bounds determine the order of processing of the subsets. These subsets are sorted by increasing sum of upper bounds of the subset's terms, and these sums, as well as the subsets ordering, do not change during query processing. Note that *critical docids* and *pivot docids* are two different concepts. Critical docids identify the docid in the docid space corresponding to a change of the dynamic pruning behavior: when a critical docid is encountered during query processing, a subset of *all* the docids greater than the critical docid will be processed in conjunctive mode. Differently, when a pivot docid is encountered during WAND processing, that *single* docid will be processed in conjunctive mode for some terms, and in disjunctive mode for some other terms.

Finally, the BMW dynamic pruning strategy applies the core processing of WAND to a new data structure built upon the posting lists of the inverted index, called block-max index. Each posting list is sequentially divided into blocks of a given number of consecutive postings, and a block upper bound is computed for each block, storing the maximum contribution of the postings in the block. When processing a query, the WAND strategy is applied to the posting list blocks, and when a document with a candidate score greater than the current threshold is identified, the actual posting lists are used to compute the actual document score.

In the following section, we propose a data structure to estimate the number of processed postings between two consecutive docids, exploiting only information already included in the full inverted index that will be used to process the queries. That data structure, that we call an *inverted index synopsis*, aims at reproducing the dynamic pruning behavior of the MaxScore, WAND and BMW strategy on the full inverted index.

5 INVERTED INDEX SYNOPSIS

We now describe a synopsis data structure for an inverted index. This data structure will support the prediction of processing times of user queries on the inverted index while minimizing or avoiding disk accesses. This synopsis data structure is substantially smaller than the original inverted index, and provides response time predictions very quickly. In Section 5.1 we describe how an inverted index synopsis is built, and its main properties, while in Section 5.2 we detail how to exploit this data structure to estimate query processing times on the original inverted index.

5.1 Synopsis

Given a document collection D and a sampling probability $0 < \gamma < 1$, we can construct a collection synopsis $\hat{D}_\gamma$ in the following way: for every document d in D, we sample it independently with probability γ and we add it to the collection synopsis $\hat{D}_\gamma$. In other words, a document d_i is sampled according to a Bernoulli trial X_i with probability of success γ, and the number of documents $|\hat{D}_\gamma|$ in the collection synopsis $\hat{D}_\gamma$ is a binomial random variable with parameters $(|D|, \gamma)$. Since all documents are sampled independently, for any subset of documents $S \subseteq D$, its synopsis $\hat{S}_\gamma$ with probability γ is again a binomial random variable with parameters $(|S|, \gamma)$. As a consequence, given any subset of documents $S \subseteq D$, the number of documents in the corresponding sample $\hat{S}_\gamma \subseteq \hat{D}_\gamma$ is a random variable $|\hat{S}_\gamma| = \sum_{d_i \in S} X_i$ with expectation $E(|\hat{S}_\gamma|)$ equal to $\gamma|S|$. As a consequence, the quantity $|\hat{S}_\gamma|/\gamma$ can be used as an estimator for $|S|$.

For a synopsis $\hat{S}_\gamma$, with $\gamma > 0$, on a subset of documents $S \neq \emptyset$, standard Chernoff bounds guarantee that the synopsis size $|\hat{S}_\gamma|$ is well concentrated around its expectation $\gamma|S|$, or, equivalently, that the estimated size of the subset of documents $|\hat{S}_\gamma|/\gamma$ is well concentrated around its actual size $|S|$, provided that $|\hat{S}_\gamma| \neq \emptyset$. In fact, by direct application of the two-sided Chernoff bound[2], the probability that the synopsis size $|\hat{S}_\gamma|$ deviates from $\gamma|S|$ by more than a factor $\varepsilon \geq 0$ is such that

$$\Pr\left[\left\|\frac{|\hat{S}_\gamma|}{\gamma|S|} - 1\right\| \geq \varepsilon\right] \leq 2\exp\left(-\frac{\varepsilon^2}{2+\varepsilon}\gamma|S|\right), \tag{3}$$

where $\|\cdot\|$ denotes the absolute value, and $|\cdot|$ denotes the cardinality.

If the set S represents the docids in a posting list and $\hat{S}_\gamma$ is the set of docids in the corresponding posting list built using the sampled documents, the quantity

$$r = \left\|\frac{|\hat{S}_\gamma|}{\gamma|S|} - 1\right\| \tag{4}$$

[2]https://www.cs.princeton.edu/courses/archive/fall09/cos521/Handouts/probabilityandcomputing.pdf

represents the *relative error* between the actual posting list size $|S|$ and its estimation $|\hat{S}_\gamma|/\gamma$, obtained using the sampled posting list size. In fact, given an inverted index on D and a term t_i, its posting list $I_i(D)$ is a set of documents, i.e., $I_i \subseteq D$. From a collection synopsis $\hat{D}_\gamma$, we can build an inverted index on its sampled documents, and the posting list $I_i(\hat{D}_\gamma)$ corresponds with the synopsis of the posting lists of the original collection $\hat{I}_{\gamma,i}(D)$, i.e., $I_i(\hat{D}_\gamma) = \hat{I}_{\gamma,i}(D)$. Since any union/intersection of posting lists is a subset of documents in the collection, the inverted index synopsis can be used to estimate the expected number of documents processed in any query, processed either in OR mode (i.e., union of posting lists) or in AND mode (i.e., intersection of posting lists):

$$E\big(|\hat{I}_{\gamma,1}(D) \cup \ldots \cup \hat{I}_{\gamma,n}(D)|\big) = \gamma |I_1(D) \cup \ldots \cup I_n(D)|, \tag{5}$$

$$E\big(|\hat{I}_{\gamma,1}(D) \cap \ldots \cap \hat{I}_{\gamma,n}(D)|\big) = \gamma |I_1(D) \cap \ldots \cap I_n(D)|. \tag{6}$$

Hence, the inverted index synopsis can be used to estimate, on average, the size of unions and/or intersection of postings lists or their parts, limited by the same initial and final docid. As we discussed in Section 4, MaxScore, WAND and BMW involve unions and intersections of parts of posting lists delimited by critical docids. Hence, an index synopsis can estimate the number of processed postings during query processing for any of such strategies.

When generating an index synopsis, two approaches may be used to assign docids: (i) in the synopsis we store the *original* docids assigned to the documents when the collection was indexed, or (ii) in the synopsis we store the *remapped* docids, assigning to the documents new docids after the collection sampling. Both approaches have some benefits. In the *original* approach, we can leverage the knowledge of the original docid to quickly identify the corresponding document in a direct index, for example to implements query expansion techniques. In the *remapped* approach, the gaps between to docids in the same posting lists are expected to be smaller than in the *original* approach, leading to greater savings in posting lists compression. We evaluate the benefits of both approaches in Section 8.

5.2 Query Efficiency Prediction using Synopses

The processing time of a query against an inverted index with dynamic pruning strategies is determined by the number of posting lists being processed and the number of processed postings [23]. The number of documents processed in any union of posting lists (processed in OR mode), as well as in any intersection of posting lists (processed in AND mode), can be estimated using our inverted index synopsis. Hence, by applying the same dynamic pruning technique for the same query on the synopsis index, we can gain insights into the pruning difficulty of the query, as observed on the synopsis index by measuring the synopsis processing time, and therefore accurately estimate the query processing time on the original index.

However, in order to obtain accurate estimates, we need the pivot docids involved in the query processing using the synopsis index to be well-aligned to those that will be observed when processing on the full index. As discussed in Section 4, the identification of critical docids involves the query term upper bounds, hence the critical values, and the threshold value. In order to obtain correct estimations of term upper bounds in the inverted index synopsis, we need to maintain the statistical properties of the scoring function applied to the original index in the new index synopsis. Most scoring functions employ collection-level statistics, such as the average document length, and term-level statistics, such as the term's collection frequency. If the inverted index synopsis maintains the collection-level and term-level statistics of the original index, the underlying score distributions of documents in the synopsis' posting lists do not change, since we are uniformly sampling from the score distributions. Moreover, the term upper bounds do not change as well. Therefore, to identify the pivot docids on the synopsis index, we apply a "rescaling" of the threshold values while we process the query in the sampled docid space. Since the score distributions and the term upper bounds do not change, the growth of the threshold value is governed uniquely by the size of the min-heap, i.e., the number K of top ranked documents. In order to rescale the threshold values for query processing over the posting list synopsis, we process queries to retrieve the top γK ranked documents.

The query processing time observed for the synopsis index can be used as a feature (with no additional features) in a simple linear regression model to estimate the query processing time on the full index. Thus no offline features, as in [23], are required, nor any dynamic features computed during query processing, as in [20].

To summarize the process for query efficiency prediction using a synopsis index, an index synopsis must be generated offline, given a full index. Then, at query time, the index synopsis will be used first to process the query with a given dynamic pruning strategy, in order to obtain an estimate of the total number of processed postings and, more in general, the query processing behavior (e.g. term co-occurrence statistics, and document scores). In the following experiments, we demonstrate how such information obtained from an index synopsis can be used for two different use cases: firstly, query efficiency prediction, i.e. predicting the actual query processing times, as well as classifying tail response times; and, secondly, predicting the performance (effectiveness) of queries, in particular using post-retrieval query performance predictors. We also experimentally investigate the costs and benefits of the proposed index synopsis in the following sections.

6 RESEARCH QUESTIONS

In this paper, we conduct experiments to address the a number of research questions, thereby permitting a full analysis of an index synopsis.

We separate these research questions into four groups. The first two groups are concerned with analyzing how a synopsis can accurately model different properties of a larger index; the second two groups address two use cases of the synopsis indices.

In particular, the first group of research questions concerns the low-level space and time overheads of an index synopsis – namely, how compression is applied, how much space is required for a synopsis, and how much time is required to process queries on an index synopsis:

RQ1. What is the impact of γ on the compression of an index synopsis, and how should docids be represented?
RQ2. What are the space overheads of an index synopsis?
RQ3. What are the time overheads of an index synopsis?

In the second group, we analyze how well a synopsis index models the properties of a larger index, particularly the frequency of terms, and their union and intersection, as may be experienced during retrieval. This provides insight into the properties of the index that make it useful for predicting response times. In particular, we address the following research questions:

RQ4. How accurate are the estimates of the number of matching documents using ranked AND/OR retrieval obtained using an index synopsis, including compared to the analytical query efficiency prediction approach of [34]?
RQ5. How accurate are the estimates of the number of matching documents using dynamic pruning strategies for retrieval?

Next, we turn to address our key application of index synopses – the prediction of the response time of the search engine. In Section 10, we measure the extent that this can be accurately achieved, in comparison to existing approaches, in addressing two research questions, namely:

RQ6. Can our index synopses accurately predict *overall* response times for retrieval observed on larger indices?

RQ7. Can our index synopses accurately classify *tail* response times for retrieval observed on larger indices?

For our final use case, we consider another benefit of performing an initial retrieval on the index synopsis, where we can leverage the results returned by processing the query on the synopsis to compute post-retrieval query performance predictors, i.e., predicting, for each query, the likely effectiveness of the search engine for that query. We investigate the accuracy of post-retrieval query performance predictors when calculating using index synopses in Section 11, through the following research question:

RQ8. How does the correlation of query performance predictors calculated on the synopsis index compare with those calculated on the full index?

In the following, we report the experimental setup that applies to all of our experiments, before reporting empirical results for all of the aforementioned research questions. The first two groups of research questions, concerned with the overheads of a synopsis index and how a synopsis can accurately model different properties of a larger index, are addressed Sections 8 & 9, respectively. The latter two groups address the two prediction use cases of index synopses, and are addressed in Sections 10 & 11, respectively.

7 EXPERIMENTAL SETUP

Our experiments are conducted on the TREC ClueWeb09-B corpus, which contains 50 million English web pages. We index all 50M documents using the Terrier IR platform[3], removing stopwords and applying Porter stemming. We do not index position information. Docids are assigned according to their descending PageRank score[4]. Our index is compressed using Elias-Fano encoding[5]. Documents are scored using BM25[6], and hence term and block upper bounds are also calculating using BM25, with a block size of 64 postings [25].

To evaluate the speed of query processing we use the 50,000 unique queries from the TREC 2005 Efficiency Track topics. For each query we remove stopwords and apply Porter stemming, removing empty queries and single term queries.[7] The statistics of the query set are reported in Table 1.

Table 1. Query set statistics.

Number of terms						Total
2	3	4	5	6	7+	
17,187	11,167	5,459	2,077	763	209	36,862

For learning purposes, we split the query set into a training set composed by 4,000 queries to compute the linear regression models, and the remaining 1,000 queries are used for testing the response time predictions generated by the

[3]http://terrier.org
[4]In initial experiments using different docid assignments we observed similar results, and hence omit this experiment in favor of other analyses
[5]https://github.com/tonellotto/terrier-ef
[6]https://github.com/tonellotto/terrier-micro
[7]Recall that no dynamic pruning is possible for a single term query.

models. All indices are loaded in memory before processing starts. Experiments are performed on a single core of a 8-core Intel i7-7770K with 64 GiB RAM. The number of retrieved documents, K, is $1,000$.

The inverted index synopses are built with varying sampling probabilities $\gamma = 0.001, 0.005, 0.01, 0.05$. Once the document collection is sampled, the block max indexes are regenerated, with blocks of 64 postings.

8 OVERHEADS OF AN INDEX SYNOPSIS

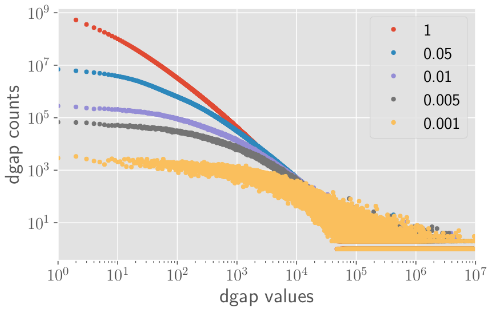
The first two research question we investigate analyze the space overhead generated by the synopsis indexes. Firstly, we compare the space overhead required to store the inverted index synopsis w.r.t. the size of the original inverted index, i.e., when $\gamma = 1$. As discussed in Section 5.1, we evaluate two approaches to assign docids in an index synopsis, namely: the *original* docids can be used, or they can be *remapped*, where new docids are assigning to the documents after the sampling.

Table 2. Synopsis index space occupancy; $\gamma = 1$ represents the full index.

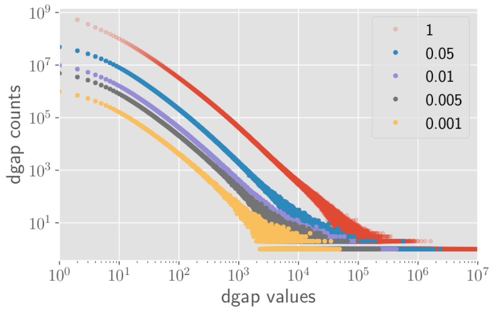
γ	Postings (M)	*original* docids		*remapped* docids	
		Space (GiB)	Reduction	Space (GiB)	Reduction
1	14,795	19.07	–	19.07	–
0.001	15	0.29	66×	0.18	106×
0.005	74	0.41	47×	0.27	71×
0.01	148	0.56	34×	0.37	52×
0.05	739	1.58	12×	1.14	17×

Table 2 reports the total number of postings (in millions), the space occupancy (in gigabytes) and the reduction factor, w.r.t. the full inverted index, of the index synopsis for different values of γ ($\gamma = 1$ denotes the full inverted index). As expected, the total number of posting in the synopsis indexes decreases almost linearly w.r.t. the sampling probability γ. Concerning the size of the synopsis indexes, there is a large reduction factor, ranging from 66× to 12× when the *original* docids are compressed with Elias-Fano, and from 106× to 17× when the *remapped* docids are compressed with Elias-Fano. As we can see, docid remapping leads to greater space reductions than using original docids. In both cases, however, the reduction factor is not linear w.r.t. γ. This is due to the fact that there are overheads in compressing posting lists, which are more apparent for short posting lists [9]. Indeed, when compressing with Elias-Fano, some bookkeeping information must be encoded at the beginning and at the end of each posting list. In fact, this effect is more apparent for smaller values of γ than for larger values of γ. For $\gamma = 0.05$, the reduction factor of the total number of postings is $14,795/739 \approx 1/\gamma \approx 20\times$, close to the 17× reduction factor for remapped docids, while for $\gamma = 0.001$, the reduction factor of the total number of postings is $14,795/15 \approx 1/\gamma \approx 993\times$, one order of magnitude greater than the 106× reduction factor for remapped docids.

We investigate the explanation behind the space reduction benefits of *remapped* docids with the help of Figures 3a and 3b. Both report the dgap distribution over the posting lists of the unique terms appearing in the 5,000 queries used in our experiments. In particular, Figure 3a illustrates the dgap distribution of the full inverted index ($\gamma = 1$) and different index synopsis when original docids are used. Large dgaps are present in all indexes, and such dgaps appear frequently in the posting lists. Conversely, as reported in Figure 3b, both counts and values of dgaps in index synopses are log-scaled w.r.t. the full index when remapped docids are compressed. For example, the number of dgaps equal to 1 is $\sim 10^9$ when $\gamma = 1$, $\sim 10^7$ when $\gamma = 0.01$ and $\sim 10^6$ when $\gamma = 0.001$. The greater number of large dgaps



(a) *Original* docids.



(b) *Remapped* docids.

Fig. 3. Distribution of dgaps for *original* (a) and *remapped* (b) docids

present when docids are not remapped has a negative impact on the compressibility of the dgaps, and this explains the better compression reduction benefits of the remapped docids w.r.t. the original docids.

In Table 3 we quantify the space overheads of synopsis indexes using *remapped* docids when used in conjunction with the regular inverted index. While in Table 2 we quantify the overheads of the synopsis w.r.t. the full index, in Table 3 we compare the overheads of the synopsis when deployed together with the full index, and to evaluate the mean impact of the synopsis on a term basis. In doing so, we are able to compare the additional storage requirements of our synopsis index w.r.t. the additional information stored by other QEP frameworks. The absolute space overhead clearly increases with the value of γ, but with a limited impact, i.e., below 2% for γ values up to 0.01. When considering how synopsis indexes relate to existing query efficiency prediction approaches, it is worth quantifying the per-term space occupancy, since such existing methods store additional statistics for each terms. The mean per-term space occupancy (in bytes) is also reported in Table 3. For γ values up to 0.01, each term, on average, occupies up to 9 bytes; the static QEP framework [23] requires 52 bytes per term to store the 13 floats corresponding to the term statistics, the analytical framework [34] requires 12 bytes per term to store the required information, while the DDS framework does not use any precomputed feature[8].

Table 3. Synopsis index space overheads.

	γ				
	1	0.001	0.005	0.01	0.05
Space (in GiB)	19.07	19.25	19.34	19.44	20.21
Overhead (in %)	0.0	0.9	1.4	1.9	6.0
Per-term space (in bytes)	0.00	6.37	7.61	8.99	19.42

Finally, in Table 4 we report the mean response times of the queries for different processing strategy, over the full index, the synopsis index, and the sum of both times (denoted Total). We also report the speedup of mean response time on the synopsis index and the overhead of the mean response time to process the queries on both indexes w.r.t. the corresponding processing on the full index, to assess the impact of different γ values. With the exception of the

[8]In Table 6 we provide further details on these QEP frameworks.

BMW strategy, all speedups w.r.t. the corresponding strategy are close to $1/\gamma$. Since the query processing times are proportional to the number of processed postings and $1/\gamma$ is the ratio between the number of postings in an original posting list and in a sample posting list, the reported speedups experimentally validate our discussion in Section 5.1, i.e., that the estimated number of postings to process is γ times the number of postings processed on the index synopsis with sampling probability γ. The speedups of BMW are smaller than the corresponding speedups of other strategies, due to the two levels of processing, i.e. on the index synopsis and on the block-max index built on top of the synopsis. The comparison of the query processing time on the full index only with that on both the synopsis and full indexes shows that the cost of prediction is limited. Indeed, these results show that the overhead is proportional to the sampling probability γ. Hence, it is possible to directly adjust the processing time overhead by setting the value of γ.

Table 4. Full index, synopsis index and total (i.e. combined synopsis and full) mean response times (in milliseconds) for the different strategies, with speedup (for synopsis) and percentage overhead (for total) w.r.t. corresponding full index processing.

		γ							
		0.001		0.005		0.01		0.05	
	Full	Syn	Total	Syn	Total	Syn	Total	Syn	Total
AND	54.3	0.06 (835×)	54.36 (+0.1%)	0.32 (170×)	54.62 (+0.6%)	0.64 (85×)	54.94 (+1.2%)	3.22 (17×)	57.52 (+5.9%)
OR	450.0	0.45 (1004×)	450.45 (+0.1%)	2.22 (202×)	452.22 (+0.5%)	4.36 (103×)	454.36 (+1.0%)	22.25 (20×)	472.25 (+4.9%)
MaxScore	87.7	0.08 (1129×)	87.78 (+0.1%)	0.40 (220×)	88.10 (+0.5%)	0.79 (111×)	88.49 (+0.9%)	4.33 (20×)	92.03 (+5.2%)
Wand	107.4	0.12 (905×)	107.52 (+0.1%)	0.61 (175×)	108.01 (+0.7%)	1.20 (90×)	108.60 (+1.1%)	6.24 (17×)	113.64 (+5.8%)
BMW	77.8	0.12 (664×)	77.92 (+0.2%)	0.60 (130×)	78.40 (+0.8%)	1.21 (65×)	79.01 (+1.6%)	6.15 (13×)	83.95 (+7.9%)

To conclude on RQ1, we observe that, the number of postings in the synopsis indexes decreases linearly with the value of γ, while the occupancy of the compressed synopsis indexes varies with the value of γ, from 106× when $\gamma = 0.001$ to 12× when $\gamma = 0.05$. In general, the smaller the value of γ, the larger the reduction factor. The best compression reductions are obtained if the docids are *remapped*, i.e., if new docids are assigned to documents after they have been sampled from the collection. The explanation for this finding is that the number of large dgaps between docids are reduced when docids are remapped w.r.t. the dgaps obtained when using the original docids.

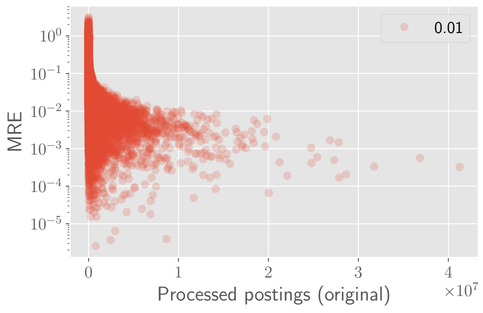
Concerning RQ2, our experiments showed that the synopsis indexes, with remapped docids, have a size of up no more than 2% of the size of the full index for γ values up to 0.01 (and 6% when $\gamma = 0.05$), and on a per-term basis, they do not require, on average, more information than existing pre-computed feature-based query efficiency frameworks.

Regarding RQ3, we can conclude that, on average, the cost of processing a query on a index synopsis is reduced by a factor equal to $1/\gamma$ w.r.t. the corresponding processing on the full index. For example, a query processed in 100 milliseconds on the full index will take only 1 millisecond to be processed on a index synopsis built from the 1% of the document in the original collection.

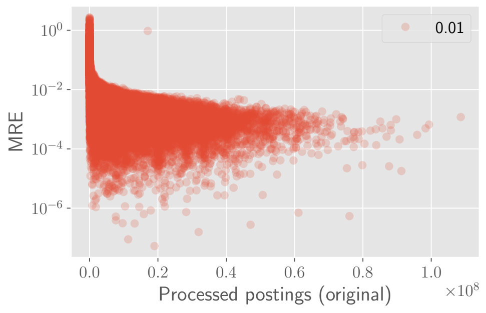
In the following sections, we use synopsis indexes with *remapped* docids in all our experiments. Next, we quantify how well the synopsis indexes exhibit the desirable properties as the full index at retrieval time.

9 INDEX SYNOPSIS PROPERTIES EVALUATION

To investigate research questions RQ4 and RQ5 on the accuracy of the estimates of the number of matching documents using ranked AND/OR retrieval and dynamic pruning strategies obtained using an index synopsis, we process the queries in conjunctive (AND) mode and disjunctive (OR) mode, as well as the MaxScore, WAND and BMW dynamic pruning strategies, on the full index and on the synopsis indexes. For all strategies we measure the number of processed



(a) Conjunctive strategy.



(b) Disjunctive strategy.

Fig. 4. Relative error distributions for conjunctive (a) and disjunctive (b) processing.

postings for each query on the full index, $|S|$, and on the synopsis index, $|\hat{S}_\gamma|$, and compute the *mean relative error* (MRE, a percentage). In particular, given a query q taken from a query log Q, we denote as $I^{(q)}$ the set of processed postings by a given strategy when processing q on the original collection, and with $\hat{I}_\gamma^{(q)})$ the set of posting processed by the same strategy on the synopsis collection. Similar to Eq. (4), we define the *query relative error* as follows:

$$r^{(q)} = \left\| \frac{|\hat{I}_\gamma^{(q)})|}{\gamma |I^{(q)}|} - 1 \right\|. \tag{7}$$

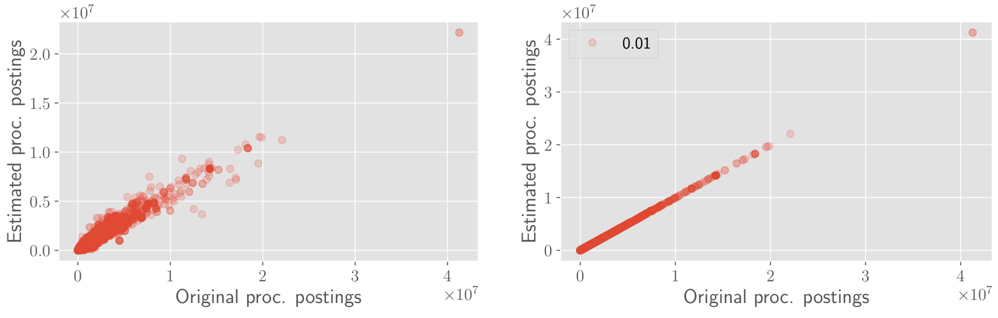
Therefore, we compute the *mean (query) relative error* MRE over a set of queries Q as:

$$\mathrm{MRE}(Q) = \frac{1}{|Q|} \sum_{q \in Q} r^{(q)}. \tag{8}$$

Table 5 reports the on the experimental query set, for different values of γ and different query processing strategies. All mean relative errors are below 50%, meaning that, on average, the estimate number of processed postings deviates from the actual number of processed postings in the range 0.5×–1.5×. For all strategies, the mean relative error decreases when γ increases. Comparing AND and OR processing modes, the mean relative error is larger for conjunctive processing than for disjunctive processing, since the size of an intersection of posting lists is larger than size of the corresponding union.

Table 5. Mean relative error (in percentage) of the number of processed postings for different values of γ and different query processing strategies.

Strategy	γ			
	0.001	0.005	0.01	0.05
AND	57.4 %	32.6 %	26.9 %	15.1 %
OR	12.2 %	8.4 %	6.8 %	3.9 %
MaxScore	27.5 %	16.6 %	13.3 %	8.8 %
WAND	33.0 %	19.6 %	15.7 %	10.8 %
BMW	45.5 %	31.9 %	28.7 %	25.2 %



(a) Analytical approximation model.

(b) Synopsis index.

Fig. 5. Estimated vs. original number of processed posting for the intersection of two term queries according to the analytical model [34] (a) and to the index synopsis with $\gamma = 0.01$ (b).

Figure 4 reports the distribution of the relative errors for conjunctive (AND) processing and disjunctive (OR) processing. Both figures report an exponential decrease of relative error values when the number of processed postings increases, as expected in Eq. (3) (note the log scale on the y axis). Note that the *maximum* relative error values decrease exponentially (with very few outliers), while for most of the reported queries, the relative error values fall well below the maximum relative error values, i.e., closer to the bottom of the graphs. For small numbers of processed postings, the relative error ranges from 300% to 0.001% for both the conjunctive and disjunctive query processing strategies. Note that the larger the number of postings to be estimated, the smaller the relative error. As the number of processed postings increases, the relative error consistently decreases below 1% for both strategies.

Recall from Section 2.1 that the analytical model [34] estimates the number of postings processed in conjunctive model for two terms according to Equation (1). Figure 5 reports the estimated vs. original number of processed posting for the intersection of two term queries according to the analytical model (a) and according to the index synopsis with $\gamma = 0.01$ (b). The analytical model estimation exhibits a 0.977 Pearson correlation, while the synopsis index has a 0.999 Pearson correlation. Moreover, the analytical model estimations are, in general, less accurate than the synopsis estimations: the former has a mean relative error of 55.7%, while the latter generates estimations very close to the actual values, with a mean relative error of 18.3%.

Next, we consider the dynamic pruning strategies in Table 5. We observe that both MaxScore and WAND obtain mean relative error values between the corresponding values for AND and OR, for all γ values (see Table 4). This is explained by the behavior of the strategies illustrated in Section 3: in both cases, most of the processed postings are part of unions and intersections of posting lists between two critical docids. The lower values for MaxScore with respect to WAND are due to the number of critical docids: having more critical docids, WAND processes more critical sections with fewer postings, and hence WAND exhibits higher relative error values on average, while MaxScore processes less critical section with more postings, with lower relative error values on average. Finally, BMW exhibits the largest deviations.

To conclude on RQ4, we observe that, on average, our synopsis indexes are able to estimate the number of processed postings on the full index with an mean relative error up to 57.4% for intersections of posting lists and up to 12.2% for unions of postings lists. When the value of γ increases, the error value decreases. In both cases, this error decreases exponentially when the size of the intersection/union to estimate increases. Our proposed estimation approach outperforms the estimation of intersection sizes proposed by [34] when $\gamma = 0.01$: their estimation based analytical modeling of query processing obtains a mean relative error of 55.7%, while our estimation based on index synopsis gets a smaller error of just 18.3%.

To conclude on RQ5, for all dynamic pruning strategies our estimations of the number of processed postings is accurate. For a typical value of γ equal to 0.01 (corresponding to a sample of 1% of the documents in the collection), the mean estimation error is 13.3% for MaxScore, 15.7% for WAND and 28.7% for BMW.

In the following section, we investigate how to use the capability of index synopsis to estimate the number of processed postings on the full index, in order to accurately estimate the processing time of queries on the full index.

10 USE CASE: PREDICTING QUERY RESPONSE TIMES

The first use case application for our index synopsis is the prediction of response times of queries. In this section we aim to exploit the index synopsis to estimate how processing queries on the full index will take, and to exploit such estimations to classify long-running queries. The prediction of query processing times can be exploited to adapt a search engine behavior at runtime according to its external operational conditions, e.g., schedule shortest queries first [23], deploy more query processing nodes if the expected processing load is increasing [14], etc. The classification of long running queries is an important optimization parameter for commercial search engines [20] since, in this context, reducing the tail latency of the long running queries is more important than reducing the mean query latency [12].

In the following, we firstly discuss the baselines we compare to, the QEP frameworks based on static features [23] (denoted as `Static` QEP) and delayed dynamic features [20] (denoted as `Dynamic` QEP). We do not compare further with the analytical model proposed by Wu et al. [34]. Indeed, the analytical model computes response time predictions by estimating the cardinality of posting lists' intersections, while our experiments reported in the previous section have shown the superiority of an index synopsis at estimating intersection sizes. Secondly, we propose and evaluate two linear regression models to estimate the query processing times on the full index leveraging our index synopsis. The first model, denoted as `Post`, performs regression on the estimated number of processed postings, obtained as described in Section 5. Moreover, we show that the query processing times on the index synopsis correlate with the query processing times on the full index. Hence our second model, denoted as `Time`, performs regression of the query processing time measured directly on the synopsis inverted index. Finally, we evaluate the performance of our proposed models leveraging index synopsis to accurately identify long running queries to reduce tail latency. Classification performance is reported in terms of precision and recall, with precision measuring how many queries are predicted as long running and with recall measuring how many long running queries are correctly classified.

10.1 Baselines & experimental setup

The two baselines which we compare to are QEP frameworks based on static features [23] (denoted as `Static` QEP) and delayed dynamic features [20] (denoted as `Dynamic` QEP). Table 6 reports a summary description of these features. The features used in the `Static` framework are the average, sum and variance of the summarized statistics for each query term.

The `Static` QEP framework exploits 39 pre-computed static features to learn a single response time prediction model *per query length*. While [23] used linear regression machine-learned models for QEP, here we use gradient-boosted regression trees models for a fair comparison with the `Dynamic` QEP. In the `Dynamic` QEP framework, every query, once received, is first executed for a given time budget (set to 10 milliseconds in our experiments), and for each query

13 features are measured during processing. For the queries that do not complete their execution within the time budget, a single response time prediction model in learned, using gradient-boosted regression trees. In contrast, in our synopsis-based QEP framework (denoted as `Synopsis`, along with the γ value used to sample the index), we train a linear regression model exploiting a single feature, namely the estimated number of processed postings or the query processing time on the synopsis index.[9]

In order to fairly compare both approaches with our response time prediction using synopsis, we build, for each query length from 2 to 6, a *local* model[10], as well as a single *global* model, considering query length as an additional feature.

For training purposes, all data sets are split into a train set containing 50% of the data and a test set containing the remaining 50% of the data. For training the gradient-boosted regression trees, the number of trees is set to 50, the learning rate to 0.1, the max depth to 3, and the loss function used is least square. In our experimental assessments on the test set, we set the prediction value to 0 if the result of a prediction is negative or missing. In the `Dynamic` QEP framework, test queries processed within the time budget are assumed to be predicted with no errors.

10.2 Linear regression exploiting the estimated number of processed postings

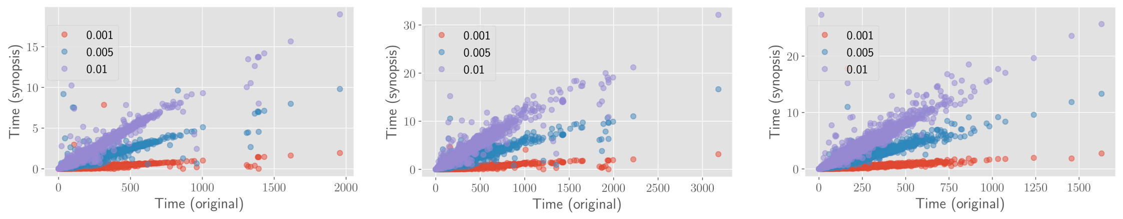
Table 6. Features used in `Static` and `Dynamic` QEP frameworks.

`Static` (max, sum and variance)	`Dynamic`
Arithmetic mean score	Minimum score
Geometric mean score	Maximum score
Harmonic mean score	Mean score
Max score	Scores variance
Variance of scores	Scores sum
Number of postings	Number of matched documents
Number of max score postings	Number of seen documents
Number of maxima postings greater than mean score	Estimated number of matched documents
Number of postings with max score	First match time
Number of postings within 5% of max score	Average match time
Number of postings within 5% of threshold	Number of matched postings
Number of promotions into top K	Number of seen postings
Inverse document frequency	Estimated number of matched postings

Table 7 reports the Pearson correlations between the estimated response times w.r.t. the original response times for the different QEP frameworks. The `Synopsis` estimations are obtained with a linear regression on only the estimated number of processed postings (i.e., the `Post` models).

We note that, for both `Static` and `Dynamic` QEP frameworks, the Pearson correlation of the local models always decreases with the number of terms, for all dynamic pruning strategies. Comparing the `Static` and `Dynamic` QEP frameworks, in most cases, the response time predictions of the `Static` QEP framework exhibit higher correlations than those of the `Dynamic` QEP framework, for all the local and global models, with some statistically significant exceptions for the BMW dynamic pruning strategy. Our results contrast with those reported in [20], i.e., `Dynamic` predictors

[9]We do not use gradient-boosted regression trees models since we deal with only a single feature.
[10]We do not use queries with 7 or more terms since they are insufficient in our training set to avoid overfitting (see Table 1).



(a) MaxScore. (b) WAND. (c) BMW.

Fig. 6. Actual response times vs. synopsis response times (both in ms) of all queries and all dynamic pruning strategies, with $\gamma = 0.001, 0.005, 0.01$.

reportedly outperformed `Static` predictors, however, we note that their experiments were conducted in a different setting – a development instance of the Bing search engine – without reporting of the particular components such as the dynamic pruning strategy in use.

Next, we consider the accuracy of the `Synopsis` predictions. Indeed, from Table 7, we observe that, for MaxScore, the `Synopsis` predictions statistically significantly outperform all local models, both `Static` and `Dynamic` , for queries with 3 terms or more, for all tested values of γ. Global models are outperformed by models using `Synopsis` for $\gamma \geq 0.005$. For WAND, the `Synopsis` predictions with $\gamma \geq 0.005$ significantly outperform the `Dynamic` predictions, but the `Static` predictions are outperformed significantly only for queries with 4+ terms. The `Static` global model is the best performing global model.

As expected from the mean relative errors reported in Table 5, the query response times for MaxScore are more accurately estimated than the query response times for WAND and BMW. In particular, we note that the proposed `Synopsis` QEP framework does not outperform the baselines when estimating the query response times obtained with the BMW dynamic pruning strategy. We obtain similar results for the MaxScore global models. However, the global models for WAND do not obtain statistically significant better correlations than both `Static` and `Dynamic`, and the global models for BMW exhibit worse correlations than both `Static` and `Dynamic`. Hence, in the following section, we investigate an alternative approach to predict response times using synopsis indexes, which exploits the query response times on the synopsis to estimate the actual response times on the full index.

10.3 Linear regression exploiting the index synopsis processing times

Alternatively, it is possible to leverage the response times of queries measured on the synopsis index to predict the response times of the queries on the full index. In fact, as shown in Figure 6, for all dynamic pruning strategies, the synopsis response times exhibit linear correlations with the actual response times.

Table 8 reports the Pearson correlations between the estimated response times w.r.t. the full index response times for the different QEP frameworks (i.e., the `Time` model). For MaxScore, the `Time` estimations correlate to the full response times similarly to the old estimations, based on the number of processed postings. In some cases, the `Time` correlations are significantly better than the corresponding `Post` ones (denoted by ▲ in Table 8). Indeed, the `Time` estimated response times correlations significantly outperform the `Post` correlations for WAND and BMW for $\gamma \geq 0.005$, for all local and global models. Note that, according to the reported numbers, the global models for MaxScore, WAND and BMW obtain correlations higher than 0.95 for $\gamma \geq 0.005$.

Table 7. Pearson correlations of estimated vs. original response time for different QEP frameworks. Estimations using synopsis use a linear regression models on the *estimated number of processed postings*. † (resp. ‡) denotes statistically significant improvements over `Static` (resp. `Dynamic`) with two-tailed Fisher r-to-z transformation tests ($p < 0.01$). The best result in each column for each dynamic pruning technique is highlighted.

Approach	Local models					Global
	2 terms	3 terms	4 terms	5 terms	6 terms	model
			MaxScore			
`Static`	0.975	0.897	0.884	0.851	0.772	0.934
`Dynamic`	0.937	0.876	0.837	0.843	0.803	0.894
`Synopsis` (0.001)	0.924	0.946†‡	0.934†‡	0.931†‡	0.903†‡	0.933‡
`Synopsis` (0.005)	0.969‡	0.972†‡	0.977†‡	0.981†‡	0.972†‡	0.969†‡
`Synopsis` (0.01)	**0.980**†‡	0.983†‡	0.984†‡	0.986†‡	0.987†‡	**0.974**†‡
`Synopsis` (0.05)	0.956‡	**0.984**†‡	**0.989**†‡	**0.991**†‡	**0.991**†‡	**0.974**†‡
			WAND			
`Static`	**0.977**	0.918	0.909	0.898	0.873	0.939
`Dynamic`	0.938	0.913	0.863	0.847	0.856	0.910
`Synopsis` (0.001)	0.917	0.900	0.874	0.873‡	0.850	0.877
`Synopsis` (0.005)	0.956‡	0.923‡	0.926†‡	0.946†‡	0.946†‡	0.907
`Synopsis` (0.01)	0.965‡	**0.935**†‡	0.934†‡	0.951†‡	0.953†‡	**0.909**
`Synopsis` (0.05)	0.944‡	**0.935**†‡	**0.943**†‡	**0.962**†‡	**0.961**†‡	0.908
			BMW			
`Static`	**0.961**	**0.953**	**0.922**	0.910	0.884	**0.941**
`Dynamic`	0.967†	0.941	0.906	**0.916**	**0.908**	0.932
`Synopsis` (0.001)	0.840	0.807	0.776	0.811	0.755	0.672
`Synopsis` (0.005)	0.901	0.841	0.835	0.891	0.872	0.727
`Synopsis` (0.01)	0.905	0.829	0.851	0.902	0.872	0.723
`Synopsis` (0.05)	0.913	0.845	0.857	0.914	0.900	0.729

Table 9 reports the mean response times (MRTs) measured of the test queries on the full index for the different dynamic pruning strategies, together with the root mean squared errors (RMSEs) of the corresponding response time predictions, computed using the global models for `Static` and `Dynamic` baselines, and `Post` and `Time Synopsis` models for different γ values. For $\gamma = 0.001$, the `Post` global models exhibit the smallest RMSE values, around 43% of the MRT for MaxScore, but approx 67% and 82% of the corresponding MRTs for WAND and BMW, respectively. However, the RMSE values are larger than the RMSE values of both `Static` and `Dynamic` global models. For larger γ values, i.e. $\gamma \geq 0.005$, all `Time` global models have smaller RMSEs than the `Post` global models, with RMSEs always lower than 30% of the corresponding MRTs.

Overall, with respect to RQ6, we conclude that our proposed index synopses accurately predict the overall response times for retrieval on larger indices. Our experiments with two linear models with a single feature, one based on the estimated number of processed postings, and the other based on the processing time of a query on the synopsis index. Both of our proposed models outperform the two state-of-the-art baselines, that exploit more complex machine-learned models using static and dynamic features, respectively (while also requiring less space – see Section 8). A synopsis index built by sampling only 1% of the documents in the full inverted index can be used to predict the response times of

Table 8. Pearson correlations of estimated vs. original response time for `Synopsis` QEP frameworks. Estimations use a linear regression models on the *query processing on the synopsis index*. † (resp. ‡) denotes statistically significant improvements over `Static` (resp. `Dynamic`) with two-tailed Fisher r-to-z transformation tests ($p < 0.01$). The best result in each column for each dynamic pruning technique is highlighted. Finally, ▲ (▼) denotes a significant improvement (degradations) w.r.t. the equivalent entry in Table 7, according to the same test.

Approach	Local models					Global
	2 terms	3 terms	4 terms	5 terms	6 terms	model
			MaxScore			
`Static`	0.975	0.897	0.884	0.851	0.772	0.934
`Dynamic`	0.937	0.876	0.837	0.843	0.803	0.894
`Synopsis` (0.001)	0.732$^{\blacktriangledown}$	0.933$^{\dagger\ddagger\blacktriangledown}$	0.669$^{\blacktriangledown}$	0.912$^{\dagger\ddagger\blacktriangledown}$	0.902†‡	0.921$^{\ddagger\blacktriangledown}$
`Synopsis` (0.005)	0.963$^{\ddagger\blacktriangledown}$	0.973†‡	0.927$^{\dagger\ddagger\blacktriangledown}$	0.957$^{\dagger\ddagger\blacktriangledown}$	0.977†‡	0.967$^{\dagger\ddagger\blacktriangledown}$
`Synopsis` (0.01)	**0.981**†‡	0.986$^{\dagger\ddagger\blacktriangle}$	0.978$^{\dagger\ddagger\blacktriangledown}$	0.989$^{\dagger\ddagger\blacktriangle}$	0.985†‡	0.982$^{\dagger\ddagger\blacktriangle}$
`Synopsis` (0.05)	0.957‡	**0.993**$^{\dagger\ddagger\blacktriangle}$	**0.992**$^{\dagger\ddagger\blacktriangle}$	**0.994**$^{\dagger\ddagger\blacktriangle}$	**0.994**$^{\dagger\ddagger\blacktriangle}$	**0.985**$^{\dagger\ddagger\blacktriangle}$
			WAND			
`Static`	**0.977**	0.918	0.909	0.898	0.873	0.939
`Dynamic`	0.938	0.913	0.863	0.847	0.856	0.910
`Synopsis` (0.001)	0.918	0.864$^{\blacktriangledown}$	0.780$^{\blacktriangledown}$	0.858	0.880	0.816$^{\blacktriangledown}$
`Synopsis` (0.005)	0.964$^{\ddagger\blacktriangle}$	0.957$^{\dagger\ddagger\blacktriangle}$	0.935$^{\dagger\ddagger\blacktriangle}$	0.959$^{\dagger\ddagger\blacktriangle}$	0.974$^{\dagger\ddagger\blacktriangle}$	0.964$^{\dagger\ddagger\blacktriangle}$
`Synopsis` (0.01)	0.976$^{\ddagger\blacktriangle}$	0.981$^{\dagger\ddagger\blacktriangle}$	0.960$^{\dagger\ddagger\blacktriangle}$	0.979$^{\dagger\ddagger\blacktriangle}$	0.985$^{\dagger\ddagger\blacktriangle}$	**0.975**$^{\dagger\ddagger\blacktriangle}$
`Synopsis` (0.05)	0.955$^{\ddagger\blacktriangle}$	**0.982**$^{\dagger\ddagger\blacktriangle}$	**0.982**$^{\dagger\ddagger\blacktriangle}$	**0.987**$^{\dagger\ddagger\blacktriangle}$	**0.989**$^{\dagger\ddagger\blacktriangle}$	**0.975**$^{\dagger\ddagger\blacktriangle}$
			BMW			
`Static`	0.961	0.953	0.922	0.910	0.884	0.941
`Dynamic`	0.967†	0.941	0.906	0.916	0.908	0.932
`Synopsis` (0.001)	0.945$^{\blacktriangle}$	0.936$^{\blacktriangle}$	0.413$^{\blacktriangledown}$	0.934$^{\dagger\ddagger\blacktriangle}$	0.931$^{\dagger\blacktriangle}$	0.715$^{\blacktriangle}$
`Synopsis` (0.005)	0.972$^{\dagger\ddagger\blacktriangle}$	0.967$^{\dagger\ddagger\blacktriangle}$	0.946$^{\dagger\ddagger\blacktriangle}$	0.971$^{\dagger\ddagger\blacktriangle}$	0.974$^{\dagger\ddagger\blacktriangle}$	0.974$^{\dagger\ddagger\blacktriangle}$
`Synopsis` (0.01)	0.928$^{\blacktriangle}$	0.973$^{\dagger\ddagger\blacktriangle}$	0.969$^{\dagger\ddagger\blacktriangle}$	0.972$^{\dagger\ddagger\blacktriangle}$	0.983$^{\dagger\ddagger\blacktriangle}$	0.980$^{\dagger\ddagger\blacktriangle}$
`Synopsis` (0.05)	**0.980**$^{\dagger\ddagger\blacktriangle}$	**0.984**$^{\dagger\ddagger\blacktriangle}$	**0.984**$^{\dagger\ddagger\blacktriangle}$	**0.988**$^{\dagger\ddagger\blacktriangle}$	**0.988**$^{\dagger\ddagger\blacktriangle}$	**0.985**$^{\dagger\ddagger\blacktriangle}$

Table 9. Mean response times (MRTs, in ms) on the original index and root mean squared errors (RMSEs) of the response time predictions using the global model on the synopsis indexes for different dynamic pruning strategy and different γ values. The best RMSE values for different γ values are in bold.

Strategy	MRT	`Static`	`Dynamic`	Synopsis RMSE			
		RMSE	RMSE	0.001	0.005	0.01	0.05
MaxScore (`Post`)	87.7	37.8	48.7	**37.0**	**25.3**	23.2	23.5
MaxScore (`Time`)				48.3	26.1	**19.7**	**17.9**
WAND (`Post`)	107.4	52.3	63.7	**71.4**	62.7	62.2	62.5
WAND (`Time`)				88.5	**39.5**	**33.0**	**33.0**
BMW (`Post`)	77.8	30.0	33.8	**65.2**	60.5	60.8	60.2
BMW (`Time`)				78.1	**20.1**	**17.6**	**15.1**

queries when dynamic pruning strategy are employed, by measuring the response time or the number of processed posting on the synopsis index, with a Pearson correlation coefficient of approx. 0.945 and a root mean square error no larger than 30% of the corresponding response time, on average.

10.4 Classification of Tail Latency Queries

Now we compare index synopsis with the baseline QEP frameworks to predict long running queries, i.e., queries with a response time greater than the 95-th percentile, also known as tail queries. We assume a query to be a tail query if its processing time exceeds the 95-th percentile of the response time distribution computed on the training queries. Since global models exhibited high correlations in our previous experiments, we focus on the analysis of the global models, for both linear regressions, namely using estimated number of processed postings or synopsis response times as feature. The precision, recall and balanced accuracy[11] values of our experiments on the test set are reported in Table 10.

Table 10. Precision, recall & balanced accuracy (in %) of tail latency classification (95-th percentile) using `Synopsis` global models (using postings, `post`, or response time, `time`, as feature) vs. `Static` and `Dynamic` QEP frameworks. † (resp. ‡) denotes statistically significant difference over `Static` (resp. `Dynamic`) according to the McNemar's test ($p < 0.01$). The best values per metrics and different γ values are in bold.

	Precision				Recall				Balanced Accuracy			
	0.001	0.005	0.01	0.05	0.001	0.005	0.01	0.05	0.001	0.005	0.01	0.05
						MaxScore						
`Static`	89.1				76.0				87.8			
`Dynamic`	89.4				54.5				77.1			
`Synopsis (Post)`	86.1‡	86.0‡	86.9†‡	87.3†‡	**77.2**‡	84.9‡	85.0†‡	85.9†‡	**88.3**‡	92.1‡	92.2†‡	92.6†‡
`Synopsis (Time)`	**96.1**†	**92.9**†‡	**93.9**†‡	**95.4**†‡	46.8†	**91.0**†‡	**95.0**†‡	**94.8**†‡	73.3†	**95.3**†‡	**97.3**†‡	**97.3**†‡
						WAND						
`Static`	88.5				75.7				87.6			
`Dynamic`	89.1				57.9				78.8			
`Synopsis (Post)`	**91.7**†	**90.8**†	**90.5**†	**90.9**†	54.0†	57.8†	56.6†	57.4†	76.9†	78.8†	78.1†	78.6†
`Synopsis (Time)`	89.7‡	87.6†‡	88.7†‡	87.5†‡	**76.7**‡	**89.9**†‡	**91.5**†‡	**92.5**†‡	**88.1**‡	**94.6**†‡	**95.5**†‡	**95.9**†‡
						BMW						
`Static`	81.2				67.7				83.4			
`Dynamic`	83.0				65.5				82.4			
`Synopsis (Post)`	55.4†‡	56.6†‡	56.9†‡	55.1†‡	24.9†‡	29.0†‡	28.0†‡	28.8†‡	61.9†‡	63.9†‡	63.5†‡	63.8†‡
`Synopsis (Time)`	**87.3**†‡	**89.0**†‡	**91.0**†‡	**90.7**†‡	**80.0**†‡	**85.2**†‡	**85.9**†‡	**88.9**†‡	**89.7**†‡	**92.3**†‡	**92.7**†‡	**94.2**†‡

On analysis of Table 10, firstly, we observe that the precision metrics for the `Dynamic` QEP framework are similar the precision results for the `Static` QEP framework. Instead, for the recall and balanced accuracy metrics, the `Static` QEP framework has higher values than the `Dynamic` QEP framework. Nevertheless, all these metrics are always outperformed by the corresponding values of the synopsis indexes for $\gamma \geq 0.005$ using postings or response time as the feature.

In general, the global models with $\gamma \geq 0.005$ relying on the synopsis response time as feature obtain better results than the corresponding global models relying on the estimated number of postings. The only exceptions are the global models applied to WAND, where the postings-based classification's precision is higher than the time-based classification's precisions. However, the recall and balanced accuracy metrics performance of the posting-based classification for WAND are lower than the recall and balanced accuracy metrics performance of the corresponding time-based models.

[11] Balanced Accuracy (BAC) is the arithmetic mean of true positive and true negative rates with a BAC of 50% indicating a random prediction.

To conclude on RQ7, our proposed synopsis index is able to classify tail response times for retrieval on larger indices with higher precision and recall than machine-learned models based on `Static` and `Dynamic` features. The global models trained on the response times on the synopsis indexes always obtain more than 0.85 in both precision and recall for $\gamma = 0.005, 0.01, 0.05$.

Overall, considering both RQ6 & RQ7, we find that use of an index synopsis allows significantly improved response time predictions compared to the baseline Static QEP approach (c.f. Tables 7 & 8) and tail query classification (Table 10) for $\gamma = 0.005$. For instance, the Global models obtained using a $\gamma = 0.005$ Synopsis index enhanced the Pearson correlation with response time by 3.5% (0.934 $\longrightarrow$ 0.967), 2.6% (0.939 $\longrightarrow$ 0.964) and 3.5% (0.941 $\longrightarrow$ 0.974) for MaxScore, WAND and BMW, respectively (higher correlations are observed for larger γ, see Table 7). Similarly, in classifying long-running tail queries, a synopsis index of $\gamma = 0.005$ can significantly improve Balanced Accuracy by 8.5% (87.8 $\longrightarrow$ 95.3), 7.9% (87.6 $\longrightarrow$ 94.6) and 10.6% (83.4 $\longrightarrow$ 92.3) for MaxScore, WAND and BMW, respectively (see Table 10). Further, we note the cost of reference to a $\gamma = 0.005$ index synopsis (which has a space overhead of 1.4%, see Table 3) is on average $0.37 - 0.56$ ms, an overhead of approx. $0.36\% - 0.75\%$ (given the response times shown in Table 4). We argue that these findings demonstrate the usefulness of our proposed synopsis methodology for obtaining accurate pre-retrieval response time predictions. Moreover, a pre-retrieval on an index synopsis has added benefits allowing other applications. In the next section, we demonstrate a further use case, query performance (effectiveness) prediction.

11 USE CASE: PREDICTING QUERY PERFORMANCE

Our final use case for an index synopsis is concerned with query performance predictors (QPPs), where performance is concerned with the *effectiveness* of the retrieval system [8]. Indeed, this contrasts with Section 10 which is concerned with query *efficiency* prediction. Indeed, query performance predictors have been studied in IR for many years, with the aim of predicting, for a given query, the likely effectiveness (as measured by an evaluation metric, such as Mean Average Precision or NDCG) of the search engine for that query.

As such, a number of query performance predictors have been proposed in the literature. Most existing works contrast *pre-retrieval* (i.e. before the query has executed) vs. *post-retrieval* predictors. We go further by clearly highlighting the information required by the predictors at the time of their computation:

- **Pre-retrieval**: These query performance predictors assume that the query has not yet started executing, and as such, the only information available is that recorded in the IR system's lexicon, such as IDF, etc [17]. Other pre-computed term-level information can be characterized in this group. For example, a query only containing very frequent query terms might not be expected to perform well [18], as might be measured by the predictor Average Inverse Collection Term Frequency.
- **Post-retrieval (retrieval scores):** Predictors belonging to this family examine the score distribution of documents retrieved for the query. At a simple level, such predictors may assume that a query with a high average score for the top retrieved documents is more likely to retrieve relevant documents.
- **Post-retrieval (document contents):** Some predictors may examine the contents of the retrieved documents. For instance, a query for which the retrieved documents are focused around a single coherent is more likely to perform well – in contrast, a query which contains only uninformative words (e.g., stopwords) that occur randomly in documents will cover various topics.
- **Post-retrieval (term co-occurrence):** Some predictors may examine the co-occurrence of query terms in the corpus – if a n-gram language model is not available for the corpus (as is typical in many search engines that

index uni-gram position by default), then such a predictor may only be computed during a pass over the inverted index posting lists. For this reason, we characterize such predictors are post-retrieval, rather than some prior work that considered them as pre-retrieval.

Of note, that in contrast to query efficiency prediction, query performance predictors that are computed after retrieval has concluded are still useful, as they predict relevance, something not available to the system. This contrasts with query efficiency prediction, which is pre-retrieval in nature, as once the query has been executed, the exact response time of the system is known. Query performance predictors after the first-stage retrieval can be used in selecting the importance of different query formulations [28], or as features within later-stage learned ranking models [22].

In this section, we first examine the accuracy of post-retrieval query performance predictors when calculating using index synopses. In particular, our focus on post-retrieval is natural, given the inexpensive cost of computing pre-retrieval predictors. Then, we compare the performance of our best post-retrieval query performance predictors on index synopsis with three pre-retrieval predictors.

We apply four specific pre-retrieval query performance predictors, three specific post-retrieval query performance predictors, and one family of post-retrieval predictors called WPM, described below.

11.1 Query performance predictors

AvIDF [18]. This pre-retrieval predictor assumes that average inverse document frequency of the terms composing a query correlates positively with average precision.

AvICTF [18]. This pre-retrieval predictor is similar to AvIDF, but the average is computed inverse collection term frequency of the query terms.

SCQ [38]. This pre-retrieval predictor models the document collection and a query as vectors, and those query vectors which are more similar to the collection vector in terms of cosine distance are considered to be easier to process.

NSCQ [38]. This pre-retrieval predictor corresponds to the SCQ score divided by the number of in-vocabulary query terms.

AvPMI [16]. A strong relationship between query terms is assumed to be indicative of a well formed query that is likely to be successful. In this predictor, the average pointwise mutual information between query terms is examined. As this requires knowledge of how many documents each pair of query terms is present in, we characterize this as a post-retrieval predictor.

EnIDF [22]. This predictor considers how many documents match all of the query terms, defined as the IDF of the size of the intersection of the postings lists for all query terms. As this requires as pass through the postings lists of a query, we consider this predictor to be post-retrieval in nature.

QueryScope [18]. Similar to EnIDF, this predictor considers the number of documents matching *any* of the query terms (rather than all of the query terms). It is defined as the $-\log$ of the size of the union of the postings lists for all query terms.

Clarity Score [11]. This predictor examines the contents of the retrieved documents, to measure the coherence of the language usage in those documents. As the contents of each document must be loaded (e.g. from a forward or direct index), this predictor can be expensive in nature. Yet its inclusion in our experiments is of interest, as it tells us how representative the results from an index synopsis can be from a content point-of-view.

WPM [26]. This is a family of predictors proposed by Roitman et al., based around the Weighted Product Model, which can generate various predictors, and subsume others in the literature. For this reason, we use it as a basis for a number of post-retrieval score based predictors. Indeed, while the authors proposed a learning framework to combine various evidences into a single discriminative predictor, we generate and evaluate various predictors. In particular, WPM is based on the premise of mean retrieval scores (up to a given rank cutoff) being a good estimator of query performance prediction. However, the retrieval scores are affected by a number of biases. Hence, they propose a number of normalization:

- **invQLen**: each document's retrieval score is normalized by the inverse of the square root of the length of the query, $\frac{1}{\sqrt{|q|}}$;
- **invDocLen**: each document's retrieval score is normalized by the inverse of the document's length, $\frac{1}{|d|}$;
- **invCS**: this adjusts all documents' score according to the corpus' similarity to the query;
- **invCd**: this capture the association strength of document with the (presumably) relevant entire corpus;
- **invSD**: this adjusts the document's score relatively to its absolute divergence from the mean score of all documents.

In our experiments, we apply WPM using each of these normalizations, while also varying the rank cutoff that the mean score is calculated to.

11.2 Experimental Setup

In the following, our aim is to assess how similar the accuracy of query performance prediction is using a synopsis index compared to the full index. Indeed, attaining effective query performance predictions on a synopsis index could allow the retrieval strategy on the full index to be adjusted – for instance, query that are predicted to be ineffective may not benefit in pseudo-relevance feedback query expansion [1].

Our experiments use the same synopsis indices computed on ClueWeb09B, however due to the need for known effectiveness values, we use 200 TREC Web 2009-2012 track topics that have relevance assessments. Therefore, the query performance prediction task is defined as predicting the relative effectiveness of the IR system, which uses BM25. We measure query effectiveness using Normalised Discounted Cumulative Gain at short and long cutoffs (NDCG@20 and NDCG@1000) and Average Precision (AP@1000). Query performance prediction can then be quantified using correlations measures, such as Kendall's τ. In the following, we quantify both the correlation of the query performance predictors with the actual NDCG@1000 effectiveness, as well as between synopsis and full index predictors.

We experiment with a total of 21 pre- and post-retrieval predictors, listed in Table 11, which encompass all families of query performance predictors discussed above.

11.3 Results

In the following: Table 12 lists the correlation of the used query performance predictors calculated using the full index, with the effectiveness in terms of NDCG@20, NDCG@1000 and AP@1000 of BM25 of results obtained on the full

Table 11. Query performance predictors.

Predictor	Family
AvIDF	Pre-retrieval
AvICTF	Pre-retrieval
SCQ	Pre-retrieval
NSCQ	Pre-retrieval
AvPMI	Post-retrieval (term co-occurrence)
EnIDF	Post-retrieval (term co-occurrence)
QueryScope	Post-retrieval (term co-occurrence)
ClarityScore(*fbDocs*, *fbTerms*)	Post-retrieval (document contents)
ClarityScore(5, 50)	"
ClarityScore(20, 50)	"
ClarityScore(100, 50)	"
ClarityScore(200, 50)	"
WPM(*normalization*, *rank cutoff*)	Post-retrieval (document scores)
WPM(invQLen, 10)	"
WPM(invQLen, 100)	"
WPM(invDocLen, 10)	"
WPM(invDocLen, 100)	"
WPM(invCd, 10)	"
WPM(invCd, 100)	"
WPM(invCS, 10)	"
WPM(invCS, 100)	"
WPM(invSD, 10)	"
WPM(invSD, 100)	"

index; Table 13 lists the correlation between query performance predictors calculated on the synopsis index with those calculated on the full index.

The results in Table 12 provide an overall portrayal of the difficulty in attaining accurate query performance prediction compared to NDCG@1000 on the 200 TREC Web track queries. The highest correlation observed between predictors and the various evaluation metrics is 0.293. These low correlations exhibited are inline with the perceived difficult of query performance on web search tasks [36]. Indeed, the correlation of $\tau = 0.259$ for AP@1000 is higher than that obtained for any of the unlearned predictors reported in [36] on the same set of topics, and even higher than the deep learning-based supervised QPP technique proposed in that paper.

Among the pre-retrieval predictors in Table 12, the highest τ correlations are shown by SCQ and NSCQ – the first predictor measures the similarity between the query and the document collection while the second normalizes the SCQ score by the query length. The highest τ correlations among the post-retrieval predictors are shown by WPM(invCS, 100) and WPM(invCd, 100) – these predictors measure the average score of the top 100 retrieved documents, normalized by the query's similarity to the entire corpus and by the association strength of documents with the (presumably) relevant entire corpus, respectively. The low effectiveness of ClarityScore is also in line with the results reported in [36].

Next, Table 13 presents the observed correlations between the post-retrieval predictors[12] calculated on the synopsis indexes compared to the full index. Overall, the correlations observed vary, but are always positive, and the magnitude of the correlations increase as γ increases.

[12]We omit pre-retrieval predictors, as they would give identical results calculated on either synopsis or full index.

Table 12. Kendall's τ correlations of different QPPs w.r.t. actual NDCG@20, NDCG@1000 and AP@1000 values. Bold values denote highest correlations of pre-retrieval predictors (top) and post-retrieval predictors (bottom).

QPP	NDCG@20	NDCG@1000	AP@1000
AvIDF	0.079	0.136	0.140
AvICTF	0.071	0.126	0.133
SCQ	**0.242**	0.119	**0.181**
NSCQ	0.061	**0.137**	0.133
AvPMI	0.206	0.102	0.144
EnIDF	-0.258	-0.169	-0.240
QueryScope	-0.030	0.078	0.053
ClarityScore(5, 50)	-0.077	0.034	0.010
ClarityScore(20, 50)	-0.039	0.056	0.030
ClarityScore(100, 50)	-0.011	0.071	0.051
ClarityScore(200, 50)	-0.002	0.072	0.057
WPM(invQLen, 10)	0.198	0.195	0.229
WPM(invQLen, 100)	0.191	0.205	0.232
WPM(invDocLen, 10)	0.244	0.121	0.172
WPM(invDocLen, 100)	0.245	0.112	0.172
WPM(invCd, 10)	0.269	0.164	0.233
WPM(invCd, 100)	**0.271**	0.176	0.242
WPM(invCS, 10)	0.269	0.289	**0.259**
WPM(invCS, 100)	0.262	**0.293**	**0.259**
WPM(invSD, 10)	0.042	-0.042	-0.005
WPM(invSD, 100)	0.183	0.015	0.095

Interestingly, the term co-occurrence QPPs (AvPMI, EnIDF, QueryScope) exhibit the highest correlations ($\tau > 0.9$) – even on a synopsis index of 0.1%, the term co-occurrence information is comparable to that on a full index. This suggests that these predictors would be equally effective on the synopsis index as on the full index, although the correlations observed in Table 12 shows that they have low correlation with query effectiveness.

On the other hand, the correlations for the WPM and Clarity Score are more variable in Table 13. Clarity Score on the full index, which was shown not to be effective performance prediction in Table 12, is moderately correlated with Clarity Score on smaller synopsis indices, and this increase as γ increases; WPM varies with γ, but also with the normalization – interestingly, invCS, which was the most effective normalization for WPM in Table 12, is remarkably correlated for smaller synopsis indices (e.g., WPM(10, invCS, 1) achieves $\tau = 0.769$ for $\gamma = 0.001$ and increasing to $\tau = 0.924$ for $\gamma = 0.05$.

In Table 14 we compare the best performing pre-retrieval and post-retrieval QPPs on the full index (namely SCQ, NSCQ, WPM(invCS, 100) and WPM(invCd, 100) from Table 12) and the most correlated post-retrieval QPPs on the index synopsis(namely AvPMI, WPM(invCS, 10) and WPM(invCd, 10) from Table 13) with the effectiveness in terms of NDCG@20, NDCG@1000 and AP@1000 of BM25 results obtained on the full index. We do not consider EnIDF and QueryScope QPPs since we already identified their poor performance in Table 12.[13]

For NDCG@20, the post-retrieval QPPs on the index synopsis are not better than the pre-retrieval predictors on for $\gamma < 0.01$. However for $\gamma \geq 0.01$ and 0.05, the WPM(invCd, 100) and WPM(invCS, 10) predictors are better than SCQ.

[13]In Table 14, we do not report any statistical significance tests because the 200 TREC topics available are not enough to perform Fisher r-to-z tests on Kendall's τ correlations.

Table 13. Kendall's τ correlations of different post-retrieval QPPs on the full index w.r.t. QPPs on the synopsis index.

QPP	$\gamma = 0.001$	$\gamma = 0.005$	$\gamma = 0.01$	$\gamma = 0.05$
AvPMI	0.919	0.974	0.979	0.991
EnIDF	0.957	0.981	0.986	0.994
QueryScope	0.992	0.997	0.998	0.999
ClarityScore(5, 50)	0.337	0.382	0.438	0.526
ClarityScore(20, 50)	0.428	0.434	0.538	0.631
ClarityScore(100, 50)	0.452	0.461	0.552	0.659
ClarityScore(200, 50)	0.458	0.443	0.553	0.661
WPM(invQLen, 10)	0.421	0.651	0.732	0.881
WPM(invQLen, 100)	0.212	0.435	0.530	0.743
WPM(invDocLen, 10)	0.167	0.335	0.410	0.499
WPM(invDocLen, 100)	0.163	0.323	0.374	0.588
WPM(invCd, 10)	0.612	0.769	0.828	0.924
WPM(invCd, 100)	0.427	0.595	0.682	0.832
WPM(invCS, 10)	0.769	0.851	0.880	0.941
WPM(invCS, 100)	0.708	0.774	0.813	0.894
WPM(invSD, 10)	0.435	0.653	0.747	0.867
WPM(invSD, 100)	0.170	0.404	0.485	0.653

Next, for NDCG@1000, the WPM(invCS, 10) and WPM(invCS, 100) predictors beat the best pre-retrieval predictor, i.e., NSCQ, even with very small sampling probabilities, e.g., $\gamma = 0.001$. Indeed, the post-retrieval predictor WPM(invCS, 10) on an index synopsis of $\gamma = 0.001$ outperforms pre-retrieval predictors by 73% ($0.137 \longrightarrow 0.241$), and by 109% for $\gamma = 0.05$ ($0.137 \longrightarrow 0.287$). The same post-retrieval predictors are better then the pre-retrieval predictors also according to the AP@1000 metrics, starting with $\gamma = 0.005$. For both metrics, these post-retrieval predictors for $\gamma = 0.05$ exhibit performance comparable to those computed on the full index, confirming the results reported in Table 13.

Hence, overall in addressing RQ8, we conclude that many of the post retrieval predictors, particularly those from the recently WPM framework that normalize document scores, can be effective when calculated on very small synopsis indices, as they exhibit high correlations with the same predictors calculated on the full index. Moreover, they are more effective than the best pre-retrieval predictors, and their computation requires an almost negligible amount on time to process the query on the synopsis index. Overall, we conclude that this demonstrates value in the use of synopsis indices for query performance prediction. Indeed, our results open up new potential applications of post-retrieval query performance predictors – calculated using an index synopsis – in adjusting first-pass retrieval strategies.

12 CONCLUSIONS

In this paper, we have proposed index synopses, which are random samples of complete document indices. Our detailed empirical results have shown that our index synopses are able to reproduce the dynamic pruning behavior of the MaxScore, WAND and BMW strategies on a full inverted index. In particular, in Section 11 we showed that an index synopsis containing no more than 0.5% of the original collection can be used to obtain accurate query efficiency predictions for dynamic pruning strategies, to estimate the processing times of queries on the full index. Indeed, use of an index synopsis allows significantly improved response time predictions compared to the baseline Static QEP approach (c.f. Tables 7 & 8) and tail query classification (Table 10) for $\gamma = 0.005$. For instance, the Global models obtained using a $\gamma = 0.005$ Synopsis index enhanced the Pearson correlation with response time by 3.5% ($0.934 \longrightarrow$

Table 14. Kendall's τ correlations for selected pre-retrieval and post-retrieval predictors on synopsis and full indices. For each measure and index, we emphasize in bold the correlations for post-retrieval predictors that exceed the correlations exhibited by the pre-retrieval predictors.

QPP	$\gamma = 0.001$	$\gamma = 0.005$	$\gamma = 0.01$	$\gamma = 0.05$	$\gamma = 1$
			NDCG@20		
SCQ			0.242		
NSCQ			0.061		
AvPMI	0.186	0.204	0.208	0.206	0.206
WPM(invCd, 10)	0.156	0.216	**0.247**	**0.267**	**0.267**
WPM(invCd, 100)	0.126	0.149	0.181	0.227	0.244
WPM(invCS, 10)	0.179	0.226	0.240	0.259	0.264
WPM(invCS, 100)	0.151	0.175	0.188	0.227	0.245
			NDCG@1000		
SCQ			0.119		
NSCQ			0.137		
AvPMI	0.093	0.100	0.103	0.102	0.102
WPM(invCd, 10)	0.124	0.169	0.184	0.180	0.171
WPM(invCd, 100)	0.070	0.107	0.143	0.175	0.174
WPM(invCS, 10)	**0.241**	**0.277**	**0.283**	**0.287**	**0.285**
WPM(invCS, 100)	0.208	0.235	0.250	0.276	0.281
			AP@1000		
SCQ			0.181		
NSCQ			0.133		
AvPMI	0.131	0.141	0.145	0.144	0.144
WPM(invCd, 10)	0.153	0.219	0.241	0.244	0.238
WPM(invCd, 100)	0.091	0.139	0.181	0.228	0.232
WPM(invCS, 10)	**0.192**	**0.234**	**0.244**	**0.255**	**0.255**
WPM(invCS, 100)	0.157	0.187	0.203	0.234	0.245

0.967), 2.6% (0.939 $\longrightarrow$ 0.964) and 3.5% (0.941 $\longrightarrow$ 0.974) for MaxScore, WAND and BMW, respectively (higher correlations are observed for larger γ, see Table 7). Similarly, in classifying long-running tail queries, a synopsis index of $\gamma = 0.005$ can significantly improve Balanced Accuracy by 8.5% (87.8 $\longrightarrow$ 95.3), 7.9% (87.6 $\longrightarrow$ 94.6) and 10.6% (83.4 $\longrightarrow$ 92.3) for MaxScore, WAND and BMW, respectively (see Table 10). The cost of reference to a $\gamma = 0.005$ index synopsis (which has a space overhead of 1.4%, see Table 3) is on average 0.37 − 0.56 ms, an overhead of approx. 0.36% − 0.75% (response times are shown in Table 4).

Moreover, in Section 11 we showed that post-retrieval query performance predictors calculated on an index synopsis can outperform pre-retrieval query performance predictors and can accurately estimate the corresponding post-retrieval query performance predictors calculated on the full index. Indeed, on an index synopsis of $\gamma = 0.001$ outperforms pre-retrieval predictors by 73% (0.137 $\longrightarrow$ 0.241), and by 109% for $\gamma = 0.05$ (0.137 $\longrightarrow$ 0.287).

The benefits of both accurate query efficiency prediction and query performance prediction open new areas for possible future work, for example allowing a component that *plans* a query's execution appropriately. For instance, having accurate query performance (effectiveness) predictors might allow to recognize when query expansion might be appropriate, and, using the query efficiency prediction, when the more complex query formulation would have time to execute with an acceptable response time. This would go further than existing works using query efficiency prediction alone [24, 30].

Further, our work here has been entirely focused on making prediction for a single index. In future work, we will consider how to apply index synopses across a tiered index layout, as used by large search engines. Moreover, in future work we plan to investigate the impact of sampling in different scenarios, such as sampling at snippet/paragraph granularity to improve the efficiency of query expansion [35]. Finally, we will investigate how document/snippet sampling can be combined with a neural ranking model for the first-pass retrieval [37] to achieve efficient neural retrieval.

REFERENCES

[1] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. 2004. Query difficulty, robustness, and selective application of query expansion. In *Advances in Information Retrieval (ECIR '04)*. Springer, 127–137. https://doi.org/10.1007/978-3-540-24752-4_10

[2] Krishna Bharat and Andrei Broder. 1998. A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines. In *Proceedings of the Seventh International Conference on World Wide Web 7 (WWW7)*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 379–388. http://dl.acm.org/citation.cfm?id=297805.297863

[3] Daniele Broccolo, Craig Macdonald, Salvatore Orlando, Iadh Ounis, Raffaele Perego, Fabrizio Silvestri, and Nicola Tonellotto. 2013. Load-sensitive Selective Pruning for Distributed Search. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 379–388. https://doi.org/10.1145/2505515.2505699

[4] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management* (New Orleans, LA, USA). ACM, New York, NY, USA, 426–434. https://doi.org/10.1145/956863.956944

[5] Eric W. Brown. 1995. Fast Evaluation of Structured Queries for Information Retrieval. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '95)*. ACM, New York, NY, USA, 30–38. https://doi.org/10.1145/215206.215329

[6] Stefan Büttcher and Charles L. A. Clarke. 2006. A Document-centric Approach to Static Index Pruning in Text Retrieval Systems. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM '06)*. ACM, New York, NY, USA, 182–189. https://doi.org/10.1145/1183614.1183644

[7] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S. Maarek, and Aya Soffer. 2001. Static Index Pruning for Information Retrieval Systems. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. ACM, New York, NY, USA, 43–50. https://doi.org/10.1145/383952.383958

[8] David Carmel and Elad Yom-Tov. 2010. Estimating the Query Difficulty for Information Retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2, 1 (2010), 1–89. https://doi.org/10.2200/S00235ED1V01Y201004ICR015 arXiv:https://doi.org/10.2200/S00235ED1V01Y201004ICR015

[9] Matteo Catena, Craig Macdonald, and Iadh Ounis. 2014. *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings.* Springer International Publishing, Cham, Chapter On Inverted Index Compression for Search Engine Efficiency, 359–371. https://doi.org/10.1007/978-3-319-06028-6_30

[10] M. Catena and N. Tonellotto. 2017. Energy-Efficient Query Processing in Web Search Engines. *IEEE Transactions on Knowledge and Data Engineering* 29, 7 (2017), 1412–1425. https://doi.org/10.1109/TKDE.2017.2681279

[11] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting Query Performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*. ACM, New York, NY, USA, 299–306. https://doi.org/10.1145/564376.564429

[12] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *Commun. ACM* 56, 2 (Feb. 2013), 74–80. https://doi.org/10.1145/2408776.2408794

[13] Shuai Ding and Torsten Suel. 2011. Faster Top-k Document Retrieval Using Block-max Indexes. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. ACM, New York, NY, USA, 993–1002. https://doi.org/10.1145/2009916.2010048

[14] Ana Freire, Craig Macdonald, Nicola Tonellotto, Iadh Ounis, and Fidel Cacheda. 2014. A Self-adapting Latency/Power Tradeoff Model for Replicated Search Engines. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14)*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/2556195.2556246

[15] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Arun N. Swami. 1996. Selectivity and Cost Estimation for Joins Based on Random Sampling. *J. Comput. Syst. Sci.* 52, 3 (June 1996), 550–569. https://doi.org/10.1006/jcss.1996.0041

[16] Claudia Hauff, Leif Azzopardi, and Djoerd Hiemstra. 2009. The Combination and Evaluation of Query Performance Prediction Methods. In *Advances in Information Retrieval (ECIR '09)*. Springer Berlin Heidelberg, Berlin, Heidelberg, 301–312. https://doi.org/10.1007/978-3-642-00958-7_28

[17] Ben He and Iadh Ounis. 2004. Inferring Query Performance Using Pre-retrieval Predictors. In *String Processing and Information Retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, 43–54. https://doi.org/10.1007/978-3-540-30213-1_5

[18] Ben He and Iadh Ounis. 2006. Query Performance Prediction. *Inf. Syst.* 31, 7 (Nov. 2006), 585–594. https://doi.org/10.1016/j.is.2005.11.003

[19] Myeongjae Jeon, Saehoon Kim, Seung-won Hwang, Yuxiong He, Sameh Elnikety, Alan L. Cox, and Scott Rixner. 2014. Predictive Parallelization: Taming Tail Latencies in Web Search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 253–262. https://doi.org/10.1145/2600428.2609572

[20] Saehoon Kim, Yuxiong He, Seung-won Hwang, Sameh Elnikety, and Seungjin Choi. 2015. Delayed-Dynamic-Selective (DDS) Prediction for Reducing Extreme Tail Latency in Web Search. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. ACM, New York, NY, USA, 7–16. https://doi.org/10.1145/2684822.2685289

[21] Yubin Kim, Jamie Callan, J. Shane Culpepper, and Alistair Moffat. 2017. Efficient Distributed Selective Search. *Inf. Retr.* 20, 3 (June 2017), 221–252. https://doi.org/10.1007/s10791-016-9290-6

[22] Craig Macdonald, Rodrygo L.T. Santos, and Iadh Ounis. 2012. On the Usefulness of Query Features for Learning to Rank. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. ACM, New York, NY, USA, 2559–2562. https://doi.org/10.1145/2396761.2398691

[23] Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2012. Learning to Predict Response Times for Online Query Scheduling. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*. ACM, New York, NY, USA, 621–630. https://doi.org/10.1145/2348283.2348367

[24] Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2017. Efficient & Effective Selective Query Rewriting with Efficiency Predictions. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 495–504. https://doi.org/10.1145/3077136.3080827

[25] Antonio Mallia, Giuseppe Ottaviano, Elia Porciani, Nicola Tonellotto, and Rossano Venturini. 2017. Faster BlockMax WAND with Variable-sized Blocks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 625–634. https://doi.org/10.1145/3077136.3080780

[26] Haggai Roitman, Shai Erera, Oren Sar-Shalom, and Bar Weiner. 2017. Enhanced Mean Retrieval Score Estimation for Query Performance Prediction. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '17)*. ACM, New York, NY, USA, 35–42. https://doi.org/10.1145/3121050.3121051

[27] Mark Sanderson, Andrew Turpin, Ying Zhang, and Falk Scholer. 2012. Differences in Effectiveness across Sub-Collections. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM âĂŹ12)*. Association for Computing Machinery, New York, NY, USA, 1965âĂŞ1969. https://doi.org/10.1145/2396761.2398553

[28] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. 2011. Intent-aware Search Result Diversification. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. ACM, New York, NY, USA, 595–604. https://doi.org/10.1145/2009916.2009997

[29] Trevor Strohman, Howard Turtle, and W. Bruce Croft. 2005. Optimization Strategies for Complex Queries. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*. ACM, New York, NY, USA, 219–225. https://doi.org/10.1145/1076034.1076074

[30] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2013. Efficient and Effective Retrieval Using Selective Pruning. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM '13)*. ACM, New York, NY, USA, 63–72. https://doi.org/10.1145/2433396.2433407

[31] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2018. Efficient Query Processing for Scalable Web Search. *Foundations and Trends in Information Retrieval* 12, 4–5 (2018), 319–492. http://dx.doi.org/10.1561/1500000057

[32] Howard Turtle and James Flood. 1995. Query evaluation: strategies and optimizations. *Information Processing and Management* 31, 6 (1995), 831–850. https://doi.org/10.1016/0306-4573(95)00020-H

[33] Ellen M. Voorhees, Daniel Samarov, and Ian Soboroff. 2017. Using Replicates in Information Retrieval Evaluation. *ACM Trans. Inf. Syst.* 36, 2, Article Article 12 (Aug. 2017), 21 pages. https://doi.org/10.1145/3086701

[34] Hao Wu and Hui Fang. 2014. Analytical Performance Modeling for Top-K Query Processing. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. ACM, New York, NY, USA, 1619–1628. https://doi.org/10.1145/2661829.2661931

[35] Jinxi Xu and W. Bruce Croft. 1996. Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 4–11. https://doi.org/10.1145/243199.243202

[36] Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. 2018. Neural Query Performance Prediction Using Weak Supervision from Multiple Signals. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. ACM, New York, NY, USA, 105–114. https://doi.org/10.1145/3209978.3210041

[37] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 497–506. https://doi.org/10.1145/3269206.3271800

[38] Ying Zhao, Falk Scholer, and Yohannes Tsegay. 2008. Effective Pre-retrieval Query Performance Prediction Using Similarity and Variability Evidence. In *Advances in Information Retrieval (ECIR'08)*. Springer-Verlag, Berlin, Heidelberg, 52–64. http://dl.acm.org/citation.cfm?id=1793274.1793285