# Empirical Asymptotic Growth of Dynamic Pruning Mechanisms

Luke Gallagher
The University of Melbourne
Melbourne, Australia
luke@hypergeometric.net

Joel Mackenzie
The University of Queensland
Brisbane, Australia
joel.mackenzie@uq.edu.au

Alistair Moffat
The University of Melbourne
Melbourne, Australia
ammoffat@unimelb.edu.au

## Abstract

Document-at-a-time query processing can be accelerated through the use of dynamic pruning mechanisms. In this empirical study we measure query time as a function of three numeric and three categorical facets, and infer relationships that allow models of computation time to be established. Using different-sized subsets of three collections, three retrieval models, and three pruning techniques, we quantify the way in which all of collection size, number of documents retrieved, and query length affect query execution times. Despite variations across pruning mechanisms, we find that in combination document retrieval is linear in the collection size when combined with retrieval depth, across all categorical dimensions. Our results allow selection of query processing techniques for specific search tasks, with the choice influenced by collection size, query length, and number of documents retrieved.

## CCS Concepts

• **Information systems → Search engine architectures and scalability**.

## Keywords

Efficiency, dynamic pruning, top-$k$ retrieval.

**ACM Reference Format:**
Luke Gallagher, Joel Mackenzie, and Alistair Moffat. 2025. Empirical Asymptotic Growth of Dynamic Pruning Mechanisms. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP 2025), December 7–10, 2025, Xi'an, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3767695.3769477

## 1 Introduction

Large-scale document retrieval systems must be both effective (returning suitable documents) and efficient (doing so with low resource cost) [9, 51], with a range of trade-offs possible between those two aspects of performance [10, 11, 16, 19, 55]. Dynamic pruning methods are a critical component in that balancing act, reducing document-at-a-time ranking costs by only considering documents that potentially exceed an evolving score threshold [8, 15, 17, 34, 35, 39, 47, 52], with many enhancements having been considered [5, 14, 23, 46, 50] across a variety of settings [25, 27, 37, 41].



*SIGIR-AP 2025, Xi'an, China*

ACM ISBN 979-8-4007-2218-9/2025/12
https://doi.org/10.1145/3767695.3769477

One avenue of interest has been on understanding the behavior of dynamic pruning algorithms. For example, many prior empirical efforts have sought to quantify which dynamic pruning scheme is most appropriate across various settings (of which there are many), including compression codecs, similarity scores, index quantization, document reordering, and even stop term presence [17, 25, 37, 41]. There has also been interest in *predicting* the response time of queries under different query processing strategies [24, 26, 42, 49, 50, 53]. Online predictors can enable query processing to be accelerated in the aggregate, via least cost algorithmic selection of pruning methods [48, 50], choosing between (replicated) traversal strategies in a distributed environment [26], or improving query scheduling and load management [24].

In the majority of the previous work, alternative methods have been compared following a standard IR experimental paradigm – defined test collections, public query sets, and results provided via graphs and tables that compare query execution times across various settings. But there is another way of comparing algorithms – via the concepts of *asymptotic growth* and the "big-Oh" notation. In this work, we add an asymptotic lens to the previous style of empirical comparison, explicitly modeling the runtime of algorithms as their input parameters grow, and fitting polynomials to a large set of per-query latency observations. Our experiments employ three categorical dimensions (namely: the collection; the scoring function; and the dynamic pruning method), along with three numeric dimensions (namely: $d$, the collection size; $\ell$, the query length; and $k$, the number of documents to retrieve). Approximating performance into a numeric model allows comparison of relative growth across the three numeric dimensions, and leads to insights that support more carefully targeted selection of query processing techniques for specific tasks and purposes, with that choice influenced by collection size, query length, and number of documents being retrieved.

Section 2 covers the relevant background material in more detail. Section 3 describes the structure of the experiments that were carried out, and describes the three categorical and three numeric facets. Section 4 then presents the empirical results, including fitted equations that summarize the observed measurements and a preliminary application to *learned sparse retrieval*. Section 5 then concludes our presentation.

## 2 Document-at-a-time Query Processing

We now introduce precursor work relative to our contributions.

**Dynamic Pruning**. Fast evaluation of document scores is the primary goal in top-$k$ retrieval. Exhaustive methods diligently process the term postings in their entirety, producing a deterministic and rank-safe set of candidate documents. One drawback is that superfluous compute resources are consumed in the exhaustive

mode, and dynamic pruning techniques were developed to reduce this waste. In particular, dynamic pruning algorithms estimate an upper-bound similarity for each document prior to processing it, and only fully score documents that *might* make it into the current top-$k$, skipping those that certainly will *not*. Estimations are typically made with pre-computed term score upper-bounds forming a barrier to entry for a document to be deemed "essential" in resolving the (rank-safe) top-$k$ result list. Block-based dynamic pruning algorithms like BMW [15] also store per-block upper-bounds, allowing more finely grained decisions to be made at the cost of additional index space consumption.

**Bag-of-Words Similarity Scoring**. Term-weighted scoring methods [44] benefit from decades of efficiency innovations [13, 43] making them still today a fast and simple solution to first phase retrieval pipelines. The three scoring paradigms relevant in our work are BM25 [45], Query likelihood (QLD) [56] and Poisson-Laplace (PL2) [1]. The theme of dynamic pruning methodologies in a large portion of prior investigations revolves around the use of BM25 as the scoring function [15, 36, 49]. In opposition to the over reliance on BM25, a number of previous works [17, 23, 41] have considered the implications of deploying alternative scoring mechanisms within common dynamic pruning strategies to understand what trade-offs exist and how scoring function performance interacts across various index configurations and algorithmic optimizations that can provide additive efficiency improvements [25, 32].

Beyond statistical similarity scoring, *learned sparse retrieval* methods – those derived from pre-trained language models – are rapidly becoming the de facto standard for inverted index-based retrieval [11, 38, 40]. However, learned sparse retrieval methods are also known to disrupt traditional dynamic pruning algorithms [29, 33], as the contextual assignment of term weights means that long postings lists can receive high upper-bound scores – a relationship at odds with the inverse document frequency weightings that apply in most statistical rankers.

**Latency, Performance Analysis and Prediction**. Existing work has explored per-query latency as an avenue to understand performance motivated by the goal of attaining empirically backed predictive models for a range of applications in the search stack [7, 24, 26, 42, 49, 50, 53]. Typically, dynamic pruning prediction mechanisms must be fast and cheap as alternative index arrangements and *anytime* traversal methods may be used in their place [21, 30]. In distributed search, Macdonald et al. [24] analyze MaxScore and WAND performance for latency prediction in query scheduling to yield better throughput in replicated environments. Notably, they exclude single-term queries, presumably due to the non-blockwise oriented pruning methods used.

In multi-stage search systems, per-query selection mechanisms in the first phase can save on resource use for the precision oriented re-ranking phase, with coarse non-rank-safe pruning as in Tonellotto et al. [50] for example. Under a different lens, Wu and Fang [53] analyze performance of (again non-block oriented) pruning methods, formulating a model of execution time that can be described asymptotically. It is in part from this work that we draw inspiration for deriving our own estimation of time in Section 4. Straddling distributed and multi-stage search, Mackenzie et al. [26] address tail latency by dynamic prediction of an "effective" depth $k$ using reference rankings as a relevance-agnostic effectiveness measure and in replicated environments the traversal strategy can be determined to curb latency in the tail. Acceleration of pruning methods by preempting an upper-bound score threshold per-query, in essence warily (that is, an asymmetric loss function) avoids some of the score processing of (the initial) non-essential items that are otherwise required during threshold ascent (stabilization) early on [42]. Other work proposes shifting to an "index synopsis" view for predictive metadata in terms of efficiency and effectiveness [49].

## 3 Dynamic Pruning and Query Evaluation Time

We first describe the experimental design and data collection processes, followed by a discussion of the experimental results.

**Similarity Scoring and Pruning Methods**. We consider three scoring functions for ranked retrieval: BM25 [45], a well-known reference, taking $k_1 = 0.9$ and $b = 0.4$; QLD [56] with Dirichlet smoothing $\mu = 1000$; and PL2 using document length normalization and correction parameter $c = 7$ [1]. Three pruning methods were used with each of those three functions: MaxScore [52]; WAND [8]; and BMW [15]. All implementations were from the PISA search system [36]. Both the scoring function and pruning method are categorical facets in our analysis.

**Document Collections and Subcollections**. The third categorical facet is document collection. The MSMARCO-v1 [3], Gov2 [12], and CC-News [28] document collections were used, containing 8.8M passages, 25.1M documents and 43.5M documents respectively.

The documents in each collection were randomly permuted, and subsets formed by taking prefixes of size $d$, with $d$ one of the three numeric facets. That is, each smaller subset was contained within each larger subset for each collection. We took $d \in \{1 \times 10^5,\ 2 \times 10^5,\ 4 \times 10^5, 1 \times 10^6, \ldots |\mathcal{D}|\}$, forming seven indexes for MSMARCO-v1, nine for Gov2, and ten for CC-News.

**Query Length**. The second numeric facet is query length, grouped by the number of terms, $\ell$. Five sets of 1000 queries were made for each collection, one of each query length $\ell \in \{1, 2, 3, 4, 5\}$, fifteen sets in total. Queries were formed by selecting a random document from the corresponding $d = 10^5$ collection, and then sampling $\ell$ distinct non-stopword terms. This approach guarantees that every query has at least one conjunctive match in every subcollection it was applied to.

**Retrieved Results**. The final numeric facet is $k$, the number of highest-scoring documents to be returned. Typical experimental methodologies explore shallow (to provide to a user) or deep (to pass to a second refinement phase) requests; here we use $k \in \{10^1, 10^2, 10^3, 10^4\}$ to cover the entirety of that spectrum.

**Hardware and Software**. Experiments were conducted on a Linux server with two Intel Xeon Gold 6144 CPUs @ 3.5 GHz and 512 GiB main memory; PISA [36] was compiled using GCC 8.4.0. Collection indexes were built using Anserini 0.36.1 [54] with Porter stemming and stopword removal. These indexes were converted to the common index file format [22] and then encoded with SIMD-BP128 [18]. For BMW, a fixed block size of 40 was used [35]. Individual queries were measured as integral "clock tick" start and stop times in microseconds, with a query that started at tick $t_0$ and

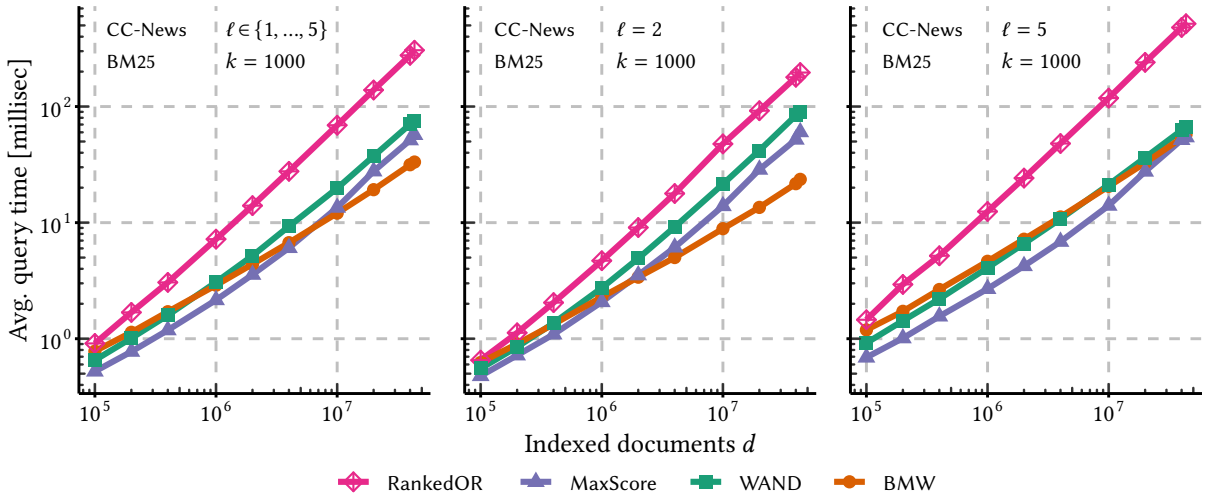


**Figure 1:** Average query time as a function of collection size for the ten CC-News subset collections (BM25, depth $k = 1000$). Left pane shows all query lengths; middle pane isolates two-term queries; and the right pane isolates five-term queries. Note that both axes are logarithmic.

ending at tick $t_1$ deemed to take $t_1 - t_0 + 1$ µs. This approach rounds up the true execution time, and ensures that no queries get reported as taking zero time.

**Indicative Results**. With 7 + 9 + 10 subcollections of differing sizes $d$, three similarity scoring protocols, four pruning strategies (including "none"), 1000 queries of each of five query lengths $\ell$, and four retrieval depths $k$, we executed 6,240,000 queries in total.

Figure 1 shows three slices through the resulting data, covering three pruning methods plus exhaustive (RankedOR) evaluation for subcollections of CC-News, retrieving $k = 1000$ documents per query using the BM25 similarity mechanism. The data slice plotted in the left pane combines $\ell \in \{1, \ldots, 5\}$ and each point represents 5000 queries. In contrast, the center and right panes restrict query length $\ell$ to two of those values, and each plotted point is thus the average time for 1000 queries of length $\ell$ over an index of size $d$. The three graphs are discussed in more detail in Section 4.

What is important to note is that both of the axes in each pane in Figure 1 are logarithmic. Straight lines thus indicate polynomial relationships, with the gradient corresponding to the polynomial degree. The goal of this investigation is to quantify the numeric exponents associated with those polynomials.

## 4 Modeling Execution Time

We now describe the insights we gleaned from our extensive set of query timings.

**Execution-Time Models**. For each combination of the three categorical facets (collection; similarity mechanism; pruning mechanism) we model the three numeric facets (collection size, $d$; query length, $\ell$; documents retrieved, $k$) via:

$$T = c_0 + c_1 \cdot d^x \cdot \ell^y \cdot k^z\,, \tag{1}$$

with $c_0 \geq 0$ and $c_1 > 0$, and with polynomials employed in response to the approximately linear behavior visible in the log-log plots in Figure 1. The 36 equations were derived using the `curve_fit` function from the SciPy software library, configured to perform nonlinear least squares using the *trust region interior reflective* algorithm [6]. Table 1 shows all 36 equations.

As an example, suppose that a BM25 scoring model is coupled with BMW on the Gov2 collection (fourth equation, center column), and suppose that $d = 2.5 \times 10^7$ and $k = 10$. The modeled execution time for a $\ell = 3$ term query is then:

$$\begin{aligned} 0 + 0.00212 \cdot d^{0.86} \cdot \ell^{1.10} \cdot k^{0.16} &\approx 23\,626 \text{ microseconds} \\ &= 23.6 \text{ milliseconds}\,, \end{aligned}$$

against a measured average execution time of 24.3 milliseconds.

Note that in this formulation the $c_0$ and $c_1$ constants are of secondary importance relative to the exponents associated with the three numeric facets in each equation, and it is the latter that drive the asymptotic growth rates that are the focus of our investigation. With that in mind, a number of interesting trends emerge.

**Exhaustive Processing**. First, exhaustive processing takes time linear in the number of documents (an exponent of $d$ close to one). While this is not at all surprising, it nevertheless serves as a useful corroboration of the experimental methodology. Moreover, exhaustive processing is highly *insensitive* to $k$ (exponent close to zero), since the cost of maintaining a min-heap to identify the $k$ highest scores is a tiny fraction of the overall cost.

**Collection Size Facet**. Second, note that the three dynamic pruning techniques are all sub-linear in $d$ (exponents less than one) once $k$ and $\ell$ are held constant. Adding documents to a collection does increase query times, but at a slower rate than at which documents are being added, suggesting that *efficiencies of scale accrue*. Moreover, for most of the combinations of categorical facets, BMW has the lowest exponent on $d$, confirming the increasing relative benefit of its more sophisticated dynamic pruning as collections grow larger. This relativity can be observed in Figure 1 with $\ell = 2$ (center), with the BMW line having a lower gradient than the other methods.

It is worth noting that the exponents on $d$ for MSMARCO-v1 in Table 1 tended to be smaller across-the-board than for the other

**Table 1:** Fitted polynomials over numeric factors: number of documents $d$; query length $\ell$; and retrieval depth $k$. Computed values are in microseconds per query.

| | | MSMARCO-v1 | Gov2 | CC-News |
|---|---|---|---|---|
| BM25 | RankedOR | $189 + 0.00031 \cdot d^{1.01}\ell^{1.06}k^{0.05}$ | $314 + 0.00088 \cdot d^{1.01}\ell^{1.23}k^{0.01}$ | $2 + 0.00209 \cdot d^{1.00}\ell^{1.09}k^{0.00}$ |
| | MaxScore | $26 + 0.00402 \cdot d^{0.75}\ell^{0.57}k^{0.24}$ | $533 + 0.00051 \cdot d^{0.98}\ell^{0.31}k^{0.13}$ | $331 + 0.00519 \cdot d^{0.89}\ell^{0.00}k^{0.09}$ |
| | WAND | $83 + 0.00812 \cdot d^{0.67}\ell^{0.65}k^{0.30}$ | $727 + 0.00072 \cdot d^{0.97}\ell^{0.56}k^{0.12}$ | $512 + 0.00749 \cdot d^{0.89}\ell^{0.00}k^{0.08}$ |
| | BMW | $0 + 0.01484 \cdot d^{0.62}\ell^{0.79}k^{0.32}$ | $0 + 0.00212 \cdot d^{0.86}\ell^{1.10}k^{0.16}$ | $169 + 0.03166 \cdot d^{0.64}\ell^{0.93}k^{0.25}$ |
| QLD | RankedOR | $215 + 0.00068 \cdot d^{1.00}\ell^{1.12}k^{0.02}$ | $2 + 0.00267 \cdot d^{0.99}\ell^{1.20}k^{0.00}$ | $2 + 0.00473 \cdot d^{1.00}\ell^{1.09}k^{0.00}$ |
| | MaxScore | $76 + 0.00326 \cdot d^{0.81}\ell^{0.66}k^{0.21}$ | $1 + 0.00412 \cdot d^{0.92}\ell^{0.91}k^{0.09}$ | $1 + 0.00501 \cdot d^{0.95}\ell^{0.82}k^{0.09}$ |
| | WAND | $180 + 0.00284 \cdot d^{0.80}\ell^{0.69}k^{0.25}$ | $1 + 0.00333 \cdot d^{0.94}\ell^{0.97}k^{0.09}$ | $0 + 0.00641 \cdot d^{0.95}\ell^{0.87}k^{0.08}$ |
| | BMW | $0 + 0.01838 \cdot d^{0.60}\ell^{0.66}k^{0.37}$ | $0 + 0.00439 \cdot d^{0.83}\ell^{1.42}k^{0.15}$ | $0 + 0.02156 \cdot d^{0.70}\ell^{1.45}k^{0.18}$ |
| PL2 | RankedOR | $279 + 0.00115 \cdot d^{1.00}\ell^{1.11}k^{0.01}$ | $5 + 0.00368 \cdot d^{1.00}\ell^{1.19}k^{0.00}$ | $6 + 0.00764 \cdot d^{1.00}\ell^{1.08}k^{0.00}$ |
| | MaxScore | $98 + 0.00554 \cdot d^{0.81}\ell^{0.64}k^{0.19}$ | $2 + 0.00549 \cdot d^{0.92}\ell^{0.87}k^{0.09}$ | $0 + 0.00751 \cdot d^{0.94}\ell^{0.76}k^{0.09}$ |
| | WAND | $224 + 0.00944 \cdot d^{0.73}\ell^{0.42}k^{0.30}$ | $1 + 0.00581 \cdot d^{0.92}\ell^{0.80}k^{0.09}$ | $0 + 0.00794 \cdot d^{0.94}\ell^{0.74}k^{0.10}$ |
| | BMW | $0 + 0.02780 \cdot d^{0.59}\ell^{0.57}k^{0.38}$ | $0 + 0.00680 \cdot d^{0.81}\ell^{1.21}k^{0.16}$ | $0 + 0.05112 \cdot d^{0.64}\ell^{1.21}k^{0.22}$ |

two collections, and the exponents on $k$ are correspondingly higher. One possibility is that, since MSMARCO-v1 is the smallest collection, it may not be sufficiently large to observe limiting behavior.

**Query Length Facet**. Third, and in marked contrast to the observation just made, note that among the three pruning mechanisms BMW tends to be the most sensitive to $\ell$, the query length. That suggests that BMW is comparatively more efficient on short queries than long ones, and that in environments where long queries are the norm, MaxScore or WAND should also be considered, with the constants $c_0$ and $c_1$ then also entering the picture. Note though that MSMARCO-v1 is not in agreement with this trend for the QLD and PL2 retrieval models.

The implication of query length $\ell$ regarding overhead in WAND and BMW has also been noted by previous researchers [24, 37], and is caused by the need to keep the postings cursors sorted during processing. This effect can be observed in Figure 1, with the relative position of the MaxScore line shifting between the center and right-side panes.

**Number of Documents Retrieved Facet**. The BMW approach is also slightly more sensitive to $k$, the count of documents retrieved. Both of these patterns can be attributed to the greater complexity of BMW – while it might score fewer documents than MaxScore or WAND, it spends more "meta-processing" time identifying documents that can be bypassed. This pattern is also visible in Figure 1, with (in the center and right data slices shown) MaxScore obtaining an advantage over WAND and BMW as the query length steps from $\ell = 2$ (center) to $\ell = 5$ (right).

**Combination of Collection Size and Retrieval Depth**. A fourth observation is that, irrespective of the other experimental settings, the exponents on $d$ and $k$ uniformly sum to approximately one. This interesting connection between $d$ and $k$ is best understood by considering what happens when both $d$ *and* $k$ double between one round of experiments and the next. The economies of scale referred to above that arise when only $d$ is scaled are then almost exactly eroded by the increase of $k$; the overall effect is of linear combined scaling behavior. Indeed, that the two exponents sum to near one provides further validation of the experimental methodology.

**Similarity Scoring**. Petri et al. [41] suggest that QLD is more sensitive to query length than BM25 because score contributions are normalized in QLD relative to the number of terms in the query. We observe similar conclusions from Table 1 in the Gov2 and CC-News collections where QLD presents greater dependence on query length $\ell$ across the three pruning methods when compared to the corresponding pruning counterpart for both BM25 and PL2.

The MaxScore pruning approach couples well with BM25 on all three collections, and tends to be the fastest in absolute terms across the ranges of $d$, $\ell$, and $k$ measured, gaining as $\ell$ increases, and confirming prior experimentation [37]. But that advantage does not extend to the QLD and PL2 calculations, for which BMW tends to be uniformly faster. This interesting result can be explained by the increased cost of scoring documents for both QLD and PL2 [17]. In other words, the "meta-processing" cost that BMW spends deciding whether to score a document or not is more likely to be worthwhile when the scoring function itself is expensive. Ultimately, if the behavior of a scoring function is not well understood, our experiments suggest that BMW is a good choice of pruning algorithm, especially if the index is dynamic.

**Prediction Error in Execution-Time Models**. We also investigated the fidelity of the query time predictions that arise from applying the models derived in connection with Table 1, noting that each model prediction is static for a given set of input parameters. Figure 2 quantifies prediction accuracy, fixing the three categorical facets (collection, similarity computation and pruning mechanism); fixing $k$; then taking all of the individual query times for all values of $d$ and stratifying the differences between predicted and actual by query length $\ell$, to generate kernel density plots. As can be seen, prediction accuracy is good, with the greater dispersion as $\ell$

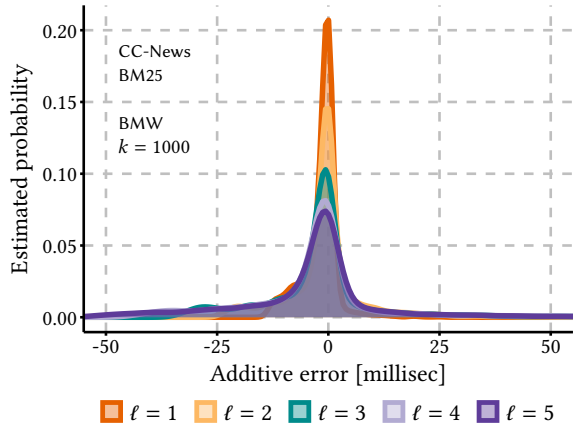


**Figure 2:** Query time prediction error distribution by query length when $k = 1000$ for row four of Table 1: BM25+BMW, for CC-News, amounting to 90 000 queries (out of 360 000 for that model).

increases a consequence of the greater query times. Similar density plots (not included here) for the Gov2 and MSMARCO-v1 collections show the same patterns.[1]

Of the three pruning mechanisms in our execution-time models, BMW results in greater prediction accuracy overall, with smaller variance in mean response time. The stratified error distributions for the other pruning approaches (also omitted here) reflect the key observations made in connection with Figure 1, with predictions for MaxScore having greater accuracy for query lengths $\ell \geq 4$, while the contrapositive was true for BMW, where prediction accuracy weakens for those same long queries.

Given the insights inferred from the execution-time models, it is plausible to deploy static predictors as a query processing diagnostic tool to inform long-term operational decisions in storage practice [20], resource utilization [24], and in the illustration of performance trade-offs framed by the (first stage) search task in support of more advanced predictors [48].

**Application to Learned Sparse Retrieval**. Our previous experiments were on three traditional similarity scoring regimes. However, methods of *learned sparse retrieval* are known to exhibit different score distributions, and in turn, problematic behavior for standard dynamic pruning algorithms [29, 33]. Our final experiment measures which – if any – of those trends translate to learned sparse retrieval mechanisms. In particular, we employ the DeeperImpact method due to both its simplicity (no query encoder is required), and its strong experimental performance [4]. We plan to extend this analysis to other learned sparse mechanisms in future work.

Following the earlier experimental setup (see Section 3), we deploy DeeperImpact on MSMARCO-v1 with a total of 560,000 queries, and model the empirical latency, with the results shown in Table 2.

Those results show that all algorithms over DeeperImpact achieve sub-linear growth with respect to $d$, the size of the collection, but with a larger exponent than the statistical scoring approaches. Similarly, a combined linear relationship exists between $d$ and $k$, and BMW continues to be most affected by increases in $k$. Compared to the previous experiments DeeperImpact is more sensitive to $\ell$,

[1]All experimental figures and data are provided as part of the code release.

**Table 2:** Fitted polynomials for DeeperImpact on the MSMARCO-v1 collection; BM25 is duplicated from Table 1.

| | | MSMARCO-v1 |
|---|---|---|
| BM25 | RankedOR | $189 + 0.00031 \cdot d^{1.01} \ell^{1.06} k^{0.05}$ |
| | MaxScore | $26 + 0.00402 \cdot d^{0.75} \ell^{0.57} k^{0.24}$ |
| | WAND | $83 + 0.00812 \cdot d^{0.67} \ell^{0.65} k^{0.30}$ |
| | BMW | $0 + 0.01484 \cdot d^{0.62} \ell^{0.79} k^{0.32}$ |
| DeeperImpact | RankedOR | $0 + 0.00390 \cdot d^{0.98} \ell^{1.24} k^{0.01}$ |
| | MaxScore | $0 + 0.00458 \cdot d^{0.90} \ell^{1.05} k^{0.13}$ |
| | WAND | $0 + 0.00424 \cdot d^{0.94} \ell^{1.16} k^{0.10}$ |
| | BMW | $0 + 0.02022 \cdot d^{0.70} \ell^{1.31} k^{0.23}$ |

suggesting one possible path for future work that might improve latency in learned sparse retrieval. Intuitively, this is a reflection of the problematic score distributions that learned sparse retrieval gives rise to – when more terms are added, it will be less likely that dynamic pruning can further accelerate query processing. In these cases the term-wise upper-bound error rates are typically much greater than those of statistical retrieval models [29].

## 5 Limitations and Conclusion

We have built models for query execution time across a set of 36 different categorical facets: three collections, three similarity scoring regimes, and four pruning strategies (the latter including "none"). We have then analyzed the resultant polynomial exponents, and discussed a range of interesting trends that emerged. Notable is that when $k$ is taken to be fixed, the different pruning methods have different asymptotic growth rates, with BMW increasing its advantage as $d$ increases; yet all methods are broadly linear if $d$ and $k$ are regarded as growing in tandem. Furthermore, we applied the same methodology to a scoring mechanism for learned sparse retrieval, and observed that pruning methods are negatively affected by the query length $\ell$ in this setting.

There are limiting factors that we have not (yet) sought to incorporate. For consistency we used a randomized document ordering to index subcollections, whereas pruned querying time can be enhanced by strategic document reordering [31]. Similarly, the compression mechanism was fixed, and it is feasible that different codecs would give rise to different trends, as document scoring may become cheaper (or also more expensive), favoring different pruning mechanisms. There are many other techniques that expedite query processing that we have not yet considered [31, 37], meaning that there is considerable scope for further work in this area. Finally, we considered only inverted index-based document-at-a-time dynamic pruning methods. Of interest is revisiting our work in algorithms specifically tailored for learned sparse retrieval [11, 38], or score-at-a-time retrieval [2, 21].

**Acknowledgment**. This work was in part funded by the Australian Research Council (Project DP200103136). The second author was supported by the Google Research Scholar program.

**Software and Data**. https://github.com/lgrz/daat-exectime.

## References


[1] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20: 357–389, 2002.

[2] V. N. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In *Proc. SIGIR*, pages 35–42, 2001.

[3] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. MS MARCO: A human generated machine reading comprehension dataset. *arXiv:1611.09268*, 2016.

[4] S. Basnet, J. Gou, A. Mallia, and T. Suel. DeeperImpact: Optimizing sparse learned index structures. In *Proc. SIGIR ReNEUIR Workshop*, 2024.

[5] E. Bortnikov, D. Carmel, and G. Golan-Gueta. Top-$k$ query processing with conditional skips. In *Proc. WWW*, pages 653–661, 2017.

[6] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM J. Sci. Comput.*, 21(1):1–23, 1999.

[7] D. Broccolo, C. Macdonald, S. Orlando, I. Ounis, R. Perego, F. Silvestri, and N. Tonellotto. Load-sensitive selective pruning for distributed search. In *Proc. CIKM*, pages 379–388, 2013.

[8] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. CIKM*, pages 426–434, 2003.

[9] S. Bruch, C. Lucchese, and F. M. Nardini. Efficient and effective tree-based and neural learning to rank. *Found. Trends Inf. Ret.*, 17(1):1–123, 2023.

[10] S. Bruch, F. M. Nardini, A. Ingber, and E. Liberty. Bridging dense and sparse maximum inner product search. *ACM Trans. Inf. Syst.*, 42(6), 2024.

[11] S. Bruch, F. M. Nardini, C. Rulli, and R. Venturini. Efficient inverted indexes for approximate retrieval over learned sparse representations. In *Proc. SIGIR*, pages 152–162, 2024.

[12] C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 terabyte track. In *Proc. TREC*, 2004.

[13] M. Crane, J. S. Culpepper, J. Lin, J. Mackenzie, and A. Trotman. A comparison of document-at-a-time and score-at-a-time query evaluation. In *Proc. WSDM*, pages 201–210, 2017.

[14] C. Dimopoulos, S. Nepomnyachiy, and T. Suel. Optimizing top-$k$ document retrieval strategies for block-max indexes. In *Proc. WSDM*, pages 113–122, 2013.

[15] S. Ding and T. Suel. Faster top-$k$ document retrieval using block-max indexes. In *Proc. SIGIR*, pages 993–1002, 2011.

[16] E. Kayaaslan, B. B. Cambazoglu, R. Blanco, F. P. Junqueira, and C. Aykanat. Energy-price-driven query processing in multi-center web search engines. In *Proc. SIGIR*, pages 983–992, 2011.

[17] O. Khattab, M. Hammoud, and T. Elsayed. Finding the best of both worlds: Faster and more robust top-$k$ document retrieval. In *Proc. SIGIR*, pages 1031–1040, 2020.

[18] D. Lemire and L. Boytsov. Decoding billions of integers per second through vectorization. *Soft. Prac. & Exp.*, 45(1):1–29, 2015.

[19] J. Leonhardt, H. Müller, K. Rudra, M. Khosla, A. Anand, and A. Anand. Efficient neural ranking using forward indexes and lightweight encoders. *ACM Trans. Inf. Syst.*, 42(5), 2024.

[20] K. Liao, A. Moffat, M. Petri, and A. Wirth. A cost model for long-term compressed data retention. In *Proc. WSDM*, pages 241–249, 2017.

[21] J. Lin and A. Trotman. Anytime ranking for impact-ordered indexes. In *Proc. ICTIR*, pages 301–304, 2015.

[22] J. Lin, J. Mackenzie, C. Kamphuis, C. Macdonald, A. Mallia, M. Siedlaczek, A. Trotman, and A. de Vries. Supporting interoperability between open-source search engines with the common index file format. In *Proc. SIGIR*, pages 2149–2152, 2020.

[23] C. Macdonald, I. Ounis, and N. Tonellotto. Upper-bound approximations for dynamic pruning. *ACM Trans. Inf. Syst.*, 29(4):17:1–17:28, 2011.

[24] C. Macdonald, N. Tonellotto, and I. Ounis. Learning to predict response times for online query scheduling. In *Proc. SIGIR*, pages 621–630, 2012.

[25] J. Mackenzie and A. Moffat. Examining the additivity of top-$k$ query processing innovations. In *Proc. CIKM*, pages 1085–1094, 2020.

[26] J. Mackenzie, J. S. Culpepper, R. Blanco, M. Crane, C. L. A. Clarke, and J. Lin. Query driven algorithm selection in early stage retrieval. In *Proc. WSDM*, pages 396–404, 2018.

[27] J. Mackenzie, C. Macdonald, F. Scholer, and J. S. Culpepper. On the cost of negation for dynamic pruning. In *Proc. ECIR*, pages 544–549, 2018.

[28] J. Mackenzie, R. Benham, M. Petri, J. R. Trippas, J. S. Culpepper, and A. Moffat. CC-News-En: A large English news corpus. In *Proc. CIKM*, pages 3077–3084, 2020.

[29] J. Mackenzie, A. Mallia, A. Moffat, and M. Petri. Accelerating learned sparse indexes via term impact decomposition. In *Proc. EMNLP Findings*, 2022.

[30] J. Mackenzie, M. Petri, and L. Gallagher. IOQP: A simple impact-ordered query processor written in rust. In *Proc. DESIRES*, pages 22–34, 2022.

[31] J. Mackenzie, M. Petri, and A. Moffat. Anytime ranking on document-ordered indexes. *ACM Trans. Inf. Syst.*, 40(1):13.1–13.32, 2022.

[32] J. Mackenzie, M. Petri, and A. Moffat. Tradeoff options for bipartite graph partitioning. *Trans. Knowledge & Data Eng.*, 35(8):8644–8657, 2023.

[33] J. Mackenzie, A. Trotman, and J. Lin. Efficient document-at-a-time and score-at-a-time query evaluation for learned sparse representations. *ACM Trans. Inf. Syst.*, 41(4), 2023.

[34] A. Mallia and E. Porciani. Faster BlockMax WAND with longer skipping. In *Proc. ECIR*, pages 771–778, 2019.

[35] A. Mallia, G. Ottaviano, E. Porciani, N. Tonellotto, and R. Venturini. Faster BlockMax WAND with variable-sized blocks. In *Proc. SIGIR*, pages 625–634, 2017.

[36] A. Mallia, M. Siedlaczek, J. Mackenzie, and T. Suel. PISA: Performant indexes and search for academia. In *Proc. SIGIR OSIRRC Workshop*, pages 50–56, 2019.

[37] A. Mallia, M. Siedlaczek, and T. Suel. An experimental study of index compression and DAAT query processing methods. In *Proc. ECIR*, pages 353–368, 2019.

[38] A. Mallia, T. Suel, and N. Tonellotto. Faster learned sparse retrieval with Block-Max pruning. In *Proc. SIGIR*, pages 2411–2415, 2024.

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

[40] T. Nguyen, S. MacAvaney, and A. Yates. A unified framework for learned sparse retrieval. In *Proc. ECIR*, pages 101–116, 2023.

[41] M. Petri, J. S. Culpepper, and A. Moffat. Exploring the magic of WAND. In *Proc. ACDS*, pages 58–65, 2013.

[42] M. Petri, A. Moffat, J. Mackenzie, J. S. Culpepper, and D. Beck. Accelerated query processing via similarity score prediction. In *Proc. SIGIR*, pages 485–494, 2019.

[43] G. E. Pibiri and R. Venturini. Techniques for inverted index compression. *ACM Comp. Surv.*, 53, 2020.

[44] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Ret.*, 3:333–389, 2009.

[45] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. TREC*, pages 109–126, 1994.

[46] D. Shan, S. Ding, J. He, H. Yan, and X. Li. Optimized top-$k$ processing with global page scores on block-max indexes. In *Proc. WSDM*, pages 423–432, 2012.

[47] T. Strohman, H. Turtle, and W. B. Croft. Optimization strategies for complex queries. In *Proc. SIGIR*, pages 219–225, 2005.

[48] G. Tolosa and A. Mallia. Many are better than one: Algorithm selection for faster top-$k$ retrieval. *Inf. Proc. & Man.*, 60(4):103359, 2023.

[49] N. Tonellotto and C. Macdonald. Using an inverted index synopsis for query latency and performance prediction. *ACM Trans. Inf. Syst.*, 38(3):29:1–29:33, 2020.

[50] N. Tonellotto, C. Macdonald, and I. Ounis. Efficient and effective retrieval using selective pruning. In *Proc. WSDM*, pages 63–72, 2013.

[51] N. Tonellotto, C. Macdonald, and I. Ounis. Efficient query processing for scalable web search. *Found. Trends Inf. Ret.*, 12:319–500, 2018.

[52] H. Turtle and J. Flood. Query evaluation: Strategies and optimizations. *Inf. Proc. & Man.*, 31(6):831–850, 1995.

[53] H. Wu and H. Fang. Analytical performance modeling for top-$k$ query processing. In *Proc. CIKM*, pages 1619–1628, 2014.

[54] P. Yang, H. Fang, and J. Lin. Anserini: Reproducible ranking baselines using Lucene. *J. Data and Inf. Quality*, 10(4):16:1–16:20, 2018.

[55] D. Yin, Y. Hu, J. Tang, T. Daly, M. Zhou, H. Ouyang, J. Chen, C. Kang, H. Deng, C. Nobata, J.-M. Langlois, and Y. Chang. Ranking relevance in Yahoo search. In *Proc. KDD*, pages 323–332, 2016.

[56] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.