Dual Skipping Guidance for Document Retrieval with Learned Sparse Representations

Yifan Qiao, Yingrui Yang, Haixin Lin, Tianbo Xiong, Xiyue Wang, Tao Yang
Department of Computer Science, University of California
Santa Barbara, California, USA

ABSTRACT
This paper proposes a dual skipping guidance scheme with hybrid scoring to accelerate document retrieval that uses learned sparse representations while still delivering a good relevance. This scheme uses both lexical BM25 and learned neural term weights to bound and compose the rank score of a candidate document separately for skipping and final ranking, and maintains two top-k thresholds during inverted index traversal. This paper evaluates time efficiency and ranking relevance of the proposed scheme in searching MS MARCO datasets.

1 INTRODUCTION
Document retrieval for searching a large dataset often uses a simple additive linear ranking to select top-k matched results before later-stage re-ranking. Dynamic pruning techniques with document skipping such as MaxScore [41], WAND [3], and Block-Max WAND (BMW) [11] can greatly speed up document retrieval by skipping low-scoring documents that are unable to appear in the final top-k list. There are various further improvements to skip more documents effectively (e.g. [4, 7, 10, 18, 19, 25, 26, 29–31, 33, 36, 40]), and these studies typically evaluate with BM25-based term weights [17]. Recently learned sparse representations have been developed to compute term weights using a neural model such as a transformer [1, 9, 12, 13, 21, 28]. Together with document expansion (e.g. [6]), document retrieval using learned sparse representations can deliver strong relevance results. A downside is that the retrieval time using learned sparse term weights is much slower than using BM25 weights as discussed in [27, 28].

The main contribution of this paper is an add-on control scheme to provide dual-threshold skipping guidance to a retrieval algorithm, and to employ a hybrid scoring with a linear combination of BM25 and learned term weights for both skipping judgment and final ranking. The evaluation in this paper with MS MARCO datasets shows that when applied to variable-sized BMW [29], the proposed scheme can deliver a very competitive relevance while its retrieval speed is close to BM25 retrieval with document expansion. Both mean response time and 95th percentile time drop significantly (e.g. varying from 1.5x to 4.3x) compared to the original baselines. Our scheme is significantly faster than a simple threshold enlarging strategy [3, 7] in reaching a similar relevance level and can leverage such a strategy for further time reduction when $k$ is relatively large.

2 BACKGROUND AND RELATED WORK
The top-k document retrieval problem identifies top ranked results in matching a query. A document representation uses a feature vector to capture the semantic of a document. If these vectors contain many zeros, then such a representation is considered sparse. For a large dataset, document retrieval often uses a simple additive formula as the first stage of search and it computes the rank score of each document $d$ as:

$$RankScore(d) = \sum_{t \in Q} w_t \cdot w_{t,d}.$$  

where $Q$ is the set of all search terms, $w_{t,d}$ is a weight contribution of term $t$ in document $d$, and $w_t$ is a document-independent or query-specific term weight. For the simplicity of presentation, assume that $w_{t,d}$ can be statically or dynamically scaled, and then we view $w_t = 1$. An example of such formula is BM25 [17] which is widely used.

For a sparse representation, a retrieval algorithm often uses an inverted index with a set of terms, and a document posting list of each term enumerating documents that contain such a term. A posting record in a posting list contains document ID and its term weight.

**Threshold-based skipping.** During the traversal of posting lists in document retrieval, the previous studies have advocated dynamic pruning strategies to skip low-scoring documents, which cannot appear on the final top-k list [3, 38]. To skip the scoring of any document, a pruning strategy computes the upper bound rank score of a candidate document $d$, called Bound$(d)$. Namely this bound value satisfies $RankScore(d) \leq Bound(d)$.

If $Bound(d) < \Theta$ where $\Theta$ is a minimum rank score of documents in the top final $k$ list, the document can be skipped. For example, WAND [3] uses the maximum term weights of documents of each posting list to determine the rank score upper bound of a pivot document while BMW [11] and its variants (e.g. [29]) optimize WAND using block-based maximum weights to compute the score upper bounds. MaxScore [41] compares the weight contribution sum upperbound of non-essential terms with the top-k threshold to guide the partitioning of query terms. In general, dynamic pruning with document skipping is often used together with the document-at-a-time or term-at-a-time traversal strategy [3, 11, 38, 41].

The above skipping is considered to be rank-safe up to $k$ in the sense that the top-$k$ documents produced are ranked correctly. Previous work has also pursued a “rank-unsafe” skipping strategy by deliberately over-estimating the current top-$k$ threshold by a factor of $F$ [3, 7, 24, 39]. There are also related strategies to obtain an accurate top-$k$ threshold earlier, e.g. [31, 33, 35, 37, 42]. While we can benefit from these studies, this paper does not study them because they represent orthogonal optimizations.

**Learned sparse representations.** Earlier sparse representation studies are conducted in [44], DeepCT [9], and SparTerm [1]. Recent work on this subject includes SPLADE [12, 13] learning token importance for document expansion with sparsity control. DeepImpact [28] learns neural term weights on documents expanded by DocT5Query [6]. Similarly, uniCOIL [21] extends the
work of COIL [14] for contextualized term weights. Document retrieval with term weights learned from a transformer has been found slow in [27, 28]. Mallia et al. [28] states that the MaxScore retrieval algorithm does not efficiently exploit the DeepImpact scores. Given “wacky weights” generated by a transformer affecting opportunities of document skipping during retrieval, Mackenzie et al. [27] advocated ranking approximation with score-at-a-time traversal.

In this paper, we still focus on document-at-a-time retrieval, and propose a complementary scheme to accelerate retrieval with dual-threshold skipping when using a learned sparse representation while our design intends to preserve or even enhance the relevance. Our skipping and final ranking adopts a hybrid formula to bound and combine rank scores based on BM25 weights and learned term weights. That is motivated by the recent studies in composing lexical and neural models in re-ranking [43] and in combining scores from sparse retrieval and dense retrieval [15, 22, 23]. We choose VBMW [29] to demonstrate our scheme because VBMW is generally acknowledged to represent the state of the art [27] for many cases. MaxScore could be a better choice for larger values of \( k \) and for long queries [30] and our technique could be applicable to MaxScore which uses threshold-based skipping, which is in our future work.

### 3 RETRIEVAL WITH DUAL GUIDANCE

Figure 1 plots the min-max scaled distribution of term weights from BM25, BM25-T5, uniCOIL, DeepImpact, and SPLADEv2 of MS MARCO passages respectively. We refer the BM25 scores calculated after DocT5Query expansion [6] as BM25-T5. This figure shows that BM25 weights and BM25-T5 weights are left skewed while the weights from all learned sparse representations are skewed to the right. An earlier study by Petri et al. [32] shows that the choice of the ranking score contribution formula and their distribution have an impact on the effectiveness of index skipping during retrieval. From that, one can conjecture that the distribution right-skewness of learned weights in all three models may be correlated to their slowness of query processing compared to BM25 and BM25-T5 with a left-skewed distribution.

As shown later in Section 4, BM25-T5 does skip more documents during retrieval compared to uniCOIL, and while a learned sparse representation performs well in terms of NDCG or MRR relevance numbers, BM25-T5 can still deliver a decent recall ratio especially with large \( k \) values. Our idea is that BM25-T5 weights can still be valuable to augment a retriever with learned weights and guide skipping. With this in mind, our design considerations are listed below.

- We treat rank scoring for document skipping differently from the final result ranking. Namely we keep two top-\( k \) rank score thresholds during index traversal. One is based on BM25 weights, another is based on the learned weights. To accomplish that, each posting record in the inverted index contains two weights for each term in a document. A retriever can maintain two queues for active top-\( k \) results based on the above two weight types.
- Bound estimation for skipping can be influenced by both BM25 and learned weights. Since we maintain two thresholds, we can use one threshold to skip based on BM25 or BM25-influenced score as discussed below, and another to remove documents that will not appear in the final top-\( k \) list.
- The above BM25-guided pruning is not rank-safe because it may skip some documents which are supposed to be included in top-\( k \) when strictly following the learned weight model. While rank-safeness is not a hard requirement, we plan to use two strategies to reduce the chance of “unsafe” skipping. One is to use a linear combination of both BM25 and learn weights in scoring to compare against a skipping threshold. In this way, the impact of BM25 weights in skipping judgment is adjustable. Since there are two top-\( k \) queues maintained, our second strategy is to study view consistency between two queues, which can improve skipping safeness. We can also adopt the threshold over-estimation strategy [3, 7, 24, 39] that enlarges a skipping threshold by a factor to improve speed as long as it does not hurt relevance.
- Our second goal is to retain or improve relevance. Since rank scores based on BM25 weights and learned weights are handily available, the final rank score can be composed by a linear combination of these two scores, which provides an opportunity of further relevance improvement with no extra cost.

![Figure 1: Distribution of BM25 and learned weights marked with distribution skewness.](image)

**Figure 1:** Distribution of BM25 and learned weights marked with distribution skewness.

![Figure 2: Dual-threshold guidance with hybrid scoring. “s” stands for skip-oriented scores, and “f” stands for final rank scores. \( \Theta \) is the top-\( k \) threshold, and \( F \) is the over-estimating factor of \( \Theta \).](image)

**Figure 2:** Dual-threshold guidance with hybrid scoring. “s” stands for skip-oriented scores, and “f” stands for final rank scores. \( \Theta \) is the top-\( k \) threshold, and \( F \) is the over-estimating factor of \( \Theta \).

Following the above discussion, we propose the following scheme with dual-threshold skipping guidance and hybrid scoring called DTHS. Figure 2 illustrates the control flow of DTHS and the rest of this section explains each component of this figure in details. Given
a retriever which uses threshold-based skipping, we extend it to use both BM25-based term weights and learned weights, compute two score upper bounds and two rank scores of a candidate document, and consults two thresholds for document skipping. The details of control guidance imposed to this retriever are described as follows.

- When the underlying retriever computes the rank score upper bound of a document for skipping judgment based on the additive formula in Section 2, it needs to be extended to compute the following two bounds. Let \( Bound_B(d) \) be the estimated maximum rank score for document \( d \) using BM25. Let \( Bound_L(d) \) be the estimated maximum rank score for \( d \) using learned weights. A linear combination of these two estimated bounds will be used as the bound for skipping judgment:

\[
Bound(d, \alpha) = \alpha Bound_B(d) + (1 - \alpha) Bound_L(d)
\]

where \( 0 \leq \alpha \leq 1 \). A large \( \alpha \) value such as 1 means the skipping condition is mainly based on BM25 weights, while a smaller \( \alpha \) value means skipping is mainly based on learned weights.

- Two rank scores are computed for each document \( d \): \( RankScore_B(d) \) and \( RankScore_L(d) \) based on BM25 weights and learned weights, respectively. We use the following linear combination as the final score of document \( d \) using parameter \( \beta \):

\[
RankScore(d, \beta) = \beta RankScore_B(d) + (1 - \beta) RankScore_L(d)
\]

where \( 0 \leq \beta \leq 1 \). If \( \beta = 1 \), the final scoring purely follows learned weights. A linear combination may boost the relevance. If \( \alpha = \beta \), it means skipping uses the same scoring formula as the final ranking to guide pruning, and the top-\( k \) retrieval algorithm is safe.

- We maintain two separate queues: Queue \( Q_x \) for the documents that have the \( k \) largest skip-oriented scores using \( RankScore(d, \alpha) \), and Queue \( Q_y \) for the documents with the \( k \) largest final rank scores using \( RankScore(d, \beta) \). Top-\( k \) threshold \( \Theta_x \) is updated based on Queue \( Q_x \), while top-\( k \) threshold \( \Theta_y \) is updated based on Queue \( Q_y \).

- We have two options of making a skipping judgment where \( F_s \) and \( F_f \) are over-estimation factors:
  - **Single-threshold skipping (ST)**: If \( Bound(d, \alpha) < F_s \Theta_x \), then scoring of document \( d \) is skipped.
  - **Dual-threshold skipping (DT)**: If \( Bound(d, \alpha) < F_s \Theta_x \) or \( Bound(d, \beta) < F_f \Theta_y \), then scoring of document \( d \) is skipped.

- When the detailed scoring of document \( d \) is not skipped, this document is added to both queues. One document is removed from each queue to maintain its size as \( k \). As shown in Figure 2, let document \( x \) be the lowest scoring document in \( Q_x \). Let document \( y \) be the lowest scoring document in \( Q_y \). If \( x = y \), we can just remove \( x \) from both queues. When \( x \neq y \), there are two options:
  - **Independent view**: The lowest-scoring document in each queue is removed separately without inter-queue coordination. This option allows different top-\( k \) documents between \( Q_x \) and \( Q_y \) to be maintained so that \( Q_x \) is more accurately matching the skipping condition regulated by \( RankScore(x, \alpha) \) formula. If removing document \( x \) from \( Q_x \) is a mistake because its relevance is actually high based on the learned weights, since such a document is still kept in \( Q_y \), this document can still appear in the final top-\( k \) list.
  - **Uniform view**: We remove \( y \) from both queues, and in this way, two queues always contain the same document sets. This design option improves the pruning safeness. Since document \( y \) will not appear in the final top-\( k \) at the end, keeping \( y \) in \( Q_x \) is unsafe. By removing \( y \) from both queues makes two queues maintain a uniform view of what should be removed and kept.

- At the end of retrieval, \( Q_f \) outputs top-\( k \) documents based on the combined rank scores \( RankScore(x, \beta) \).

### 4 Evaluation

**Setting and metrics.** Our evaluation uses the MS MARCO document and passage collections for retrieval and ranking [5, 8]. The contents in the document collections are segmented during indexing and re-grouped after retrieval using “max-passage” strategy following [34]. There are 8.8M passages with an average length of 55 words, and 3.2M documents with an average length of 1131 words. The Dev query set for passage and document ranking has 6980 and 5193 queries respectively with about one judgment label per query. Each of the passage/document ranking task of TREC Deep Learning (DL) 2019 and 2020 tracks provides a set of queries with many judgement labels per query.

In producing an inverted index, all words use lower case letters. The stand-alone BM25 and MB25-T5 indices reported in the following tables use the BERT’s Word Piece tokenizer. For learned representations, the DeepImpact index uses a tokenizer called nltk [2] while uniCOIL and SPLADEv2 use the BERT’s Word Piece tokenizer. When BM25 is used with a learned representation for DTHS, their tokenization needs to be consistent. For example, BM25 in uniCOIL/DTHS is computed for tokens based on the Word Piece tokenizer, while BM25 in DeepImpact/DTHS follows the nltk tokenizer. The index compression uses SIMD-BP128 [20], following [30]. We apply VBMW [29] with variable-sized blocks (the average block size is 1024 posting records).

Our implementation uses C++, leveraging block partitioning code from [29], and is compiled with GCC 10.2.0 and -Ofast optimization flag, running as a single thread on a Linux server with Intel i5-8259U 2.3GHz and 32GB memory.

For MS MARCO Dev set, we report the relevance in terms of mean reciprocal rank (MRR@10 on passages and MRR@100 on documents), following the official leader-board standard. One reason to choose MRR instead of using normalized discounted cumulative gain (NDCG) [16] is because such a set has about one judgment label per query, which is too sparse to use NDCG. For TREC DL test sets, we report normalized discounted cumulative gain (NDCG@10) [16], following the common practice of the previous work [12, 14, 15, 28]. NDCG is appropriate for DL test sets because they have many judgement lables per query. We also report the recall ratio which is the percentage of relevant-labeled results appeared in the final top-\( k \) results.

Before timing queries, all compressed posting lists and metadata for tested queries are pre-loaded into memory, following the same assumption in [19, 29]. Retrieval mean response times (MRT) are reported in milliseconds. The 95th percentile time (95T) is reported within parentheses in the tables below, corresponding to the time occurring in the 95th percentile and called tail latency in [25].
For all of our experiments, we perform pairwise t-test on the relevance between proposed method and corresponding baselines, no statistically significant degradation is observed at 95% confidence level.

Table 1: MRR@10 for passages and @100 for documents), NDCG@10, and mean/95th percentile time in milliseconds of different methods when \( k=1,000 \).

| Methods               | Dev | TREC DL 19 Time | TREC DL 20 NDCG | NDCG | Time |
|-----------------------|-----|-----------------|-----------------|------|------|
| BM25                  | 0.172 (74.42) | 0.425 (524.10) | 0.455 (541.17) |
| BM25-T5               | 0.277 (30.70) | 0.579 (75.16)  | 0.629 (74.19)  |
| unicOIL               | 0.347 (50132) | 0.703 (20720)  | 0.675 (204039) |
| unicOIL, F=1.7        | 0.346 (3074)  | 0.703 (73183)  | 0.675 (75214)  |
| unicOIL, F=1.9        | 0.345 (2356)  | 0.695 (55124)  | 0.672 (60172)  |
| DeepImpact            | 0.328 (71166) | 0.695 (257515) | 0.628 (243853) |
| SPLADEv2              | 0.353 (1735297) | 0.729 (1513461) | 0.714 (1408191) |
| unicOIL,DTHS          | 0.356 (3276)  | 0.707 (83169)  | 0.685 (79198)  |
| \( + F_{\alpha=1.3,F_{\beta=1}} \) | 0.353 (2437)  | 0.702 (48101)  | 0.675 (46110)  |
| DeepImpact,DTHS       | 0.344 (4596)  | 0.710 (108302) | 0.675 (113261) |
| SPLADEv2,DTHS         | 0.362 (89166) | 0.735 (1059230) | 0.714 (956207) |

MS MARCO Documents

| Methods               | MS MARCO Passages | Dev | TREC DL 19 Time | TREC DL 20 NDCG | NDCG | Time |
|-----------------------|-------------------|-----|-----------------|-----------------|------|------|
| BM25                  | 0.203 (14631)     | 0.446 (171173) | 0.451 (146422) |
| BM25-T5               | 0.303 (205435)    | 0.559 (179174) | 0.561 (174408) |
| unicOIL               | 0.353 (494136)    | 0.641 (501461) | 0.601 (512056) |
| unicOIL, F=1.5        | 0.352 (207523)    | 0.641 (201331) | 0.601 (208609) |
| unicOIL, F=1.7        | 0.351 (154348)    | 0.637 (158350) | 0.601 (153410) |
| unicOIL,DTHS          | 0.373 (259722)    | 0.676 (199459) | 0.619 (195483) |
| \( + F_{\alpha=1.3,F_{\beta=1}} \) | 0.371 (173365) | 0.670 (127203) | 0.619 (125904) |
| \( + F_{\alpha=1.5,F_{\beta=1}} \) | 0.367 (130284) | 0.669 (115240) | 0.619 (113269) |

Performance when \( k \) varies. Table 2 compares relevance and query processing time of several methods when using unicOIL weights and \( k = 10, 20, 200, \) and 1,000 for MS MARCO passages Dev set and MS MARCO documents Dev set. The 95th percentile time is reported in the parentheses next to the mean query latency number in the tables while the recall ratio is reported in the parentheses next to the mean query latency number. This recall ratio is the percentage of relevant-labeled results appeared in the final top-\( k \) results. For MS MARCO passages, MRR@10 is reported for different \( k \) values. For MS MARCO documents, MRR@100 number is reported when \( k = 200 \) and 1,000, and MRR@k number is reported when \( k = 10 \) and 20.

In general, DTHS performs well in relevance, MRT, and 95T for smaller \( k \) values with a takeaway similar as the case of \( k = 1,000 \) studied above. For example, with \( k = 10 \), DTHS performs similarly as original unicOIL in relevance while it is 1.6x and 1.4x faster in MRT and 95T on passages Dev set. For small \( k \) values such as \( k = 10 \), simple threshold over-estimation becomes ineffective for both unicOIL and DTHS because relevance drops significantly and a small enlarging factor yields limited or no time reduction. For documents Dev set, the results and observations are similar.

Result overlapping, skipping effectiveness, and design options. Table 3 compares DTHS with several baselines and impacts of its design options for DL’19 passages using unicOIL weights and \( k = 1,000 \). The default DTHS setting is listed in Row 5 and each design variation listed below changes few parameters in the default setting. Column marked “Overlap” is the percentage of overlapping documents in top-\( k \) results compared with the expected final ranking, which measures the relative rank-safeness. For BM25-T5, unicOIL with \( F = 1.7 \), and DTHS variations with \( \beta = 0 \), their result overlapping ratio listed is compared against the unicOIL final ranking results. For other design variations of DTHS with \( \beta = 0.2 \),
result overlapping is compared against DTHS with $\alpha = \beta = 0.2$, in which skipping strictly follows final rank scoring based on a linear combination of BM25 (weighted 0.2) and uniCOIL (weighted 0.8).

Column marked “BLoad” in Table 3 is the percentage of posting blocks loaded for decompression and possible scoring. Less blocks loaded imply less passages fully scored. Column marked “# Eval” is the number of fully-scored passages during retrieval. The above two numbers measure skipping effectiveness, and smaller such numbers have a good correlation with a faster response time as shown by this table. This table shows that skipping in DTHS is effective. For example, VBMW with default DTHS setting only loads 69% of posting blocks while unmodified VBMW for uniCOIL loads over 95% of blocks.

In terms of DTHS design options, hybrid scoring for final rank scores visibly improves relevance by comparing DTHS $\alpha = \beta = 0.2$ with uniCOIL ranking, and by comparing DTHS $\beta = 0$ with $\beta = 0.2$. Hybrid scoring for skip threshold comparison slightly improves pruning effectiveness by comparing $\alpha=1$, $\beta=0$ and $\alpha=0.9$, $\beta=0$. Comparing Rows 6 and 7, dual-threshold (DT) option is about 5% faster than the single threshold option while their relevance is similar. Based on the result difference between Rows 7 and 8 and between Rows 11 and 12, when two queues are managed in a uniform view, DTs final results have 99.99% overlapping with the expected ranking. Thus this option is almost rank-safe, but it reduces skipping opportunities significantly and MRT becomes much larger than that in an independent view.

## 5 CONCLUDING REMARKS

This paper proposes a dual-guidance scheme (DTHS) for document retrieval with learned sparse representations. This scheme exploits both BM25 weights and learned weights compositionally to guide skipping with dual thresholds and improve final ranking relevance. The evaluation shows DTHS effectively accelerates retrieval in mean response times and 95th percentile times while delivering a very competitive relevance. DTHS is significantly faster than a threshold enlarging strategy in reaching a similar relevance level. For relatively large $k$ values, DTHS with threshold overestimation can accelerate retrieval further. Our evaluation is reported on VBMW. The result using VBMW has a similar pattern and is not reported here. Our future work is to assess the effectiveness of dual guidance in other retrieval algorithms which use threshold-based skipping.

### REFERENCES

[1] Yang Bai, Xiaoguang Li, Gang Wang, Chao liang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Lu. 2020. SparTerm: Learning Term-based Sparse Representation for Fast Text Retrieval. ArXiv abs/2010.00676 (2020).

[2] Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. "O’Reilly Media, Inc."

[3] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Sofer, and Jason Zien. 2003. Efficient Query Evaluation Using a Two-level Retrieval Process. In Proc. of the 12th ACM International Conference on Information and Knowledge Management. 426–434.

[4] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Sofer, and Jason Zien. 2003. Efficient Query Evaluation Using a Two-Level Retrieval Process. In Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM ’03). ACM, 426–434.

[5] Daniel Fernando Campos, Tian Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Jagmudden, Li Deng, and Bhaskar Mitra. 2016. MS MARCO: A Human Generated Machine Reading COmprehension Dataset. ArXiv abs/1611.09268 (2016).

[6] David R. Chilton. 2019. From doc2query to docTTTTTquery.

[7] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trottman. 2017. A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (Washington, United Kingdom) (WSDM ’17). ACM, New York, NY, USA, 201–210.

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Fernando Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2020 Deep Learning Track. ArXiv abs/2102.07662 (2020).

[9] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020).

[10] Constantin Dimopoulou, Sergey Nepomnyashchy, and Torsten Suel. 2013. A Candidate Filtering Mechanism for Fast Top-k Query Processing on Modern CPUs. In Proc. of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. 723–732.

[11] Shuai Ding and Torsten Suel. 2011. Faster Top-k Document Retrieval Using Block-Max Indexes. In Proc. of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. 993–1002.

[12] Thibault Formal, C. Lassance, Benjamin Piwowarski, and Stéphane Cléchaint. 2021. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval. ArXiv abs/2109.10086 (2021).

[13] Thibault Formal, Benjamin Piwowarski, and Stéphane Cléchaint. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (2021).

[14] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. CoRR and NAACL. 2021 abs/2104.07186 (2021). ArXiv abs/2104.07186 https://arxiv.org/abs/2104.07186

[15] Luyu Gao, Zhuyun Dai, Tongfeng Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complementing Lexical Retrival with Semantic Residual Embedding. SCIR and arXiv:1909.01196 (2021).

[16] Kalervo Jarvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation model of information retrieval: development and comparative experiments. In Proc. of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval. 1031–1040.

[17] Karen Spärck Jones, Steve Walker, and Stephen E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. In Information Processing and Management: 779–840.

[18] Andrew Kane and Frank Wm. Tompa. 2018. Split-Lists and Initial Thresholds for WAND-Based Search. In Proc. of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval. 877–880.

[19] Omar Khatib, Mohammad Hammoud, and Tamer Elsayed. 2020. Finding the best of both worlds: Faster and more robust top-k document retrieval. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 1031–1040.

[20] Daniel Lemire and Leonid Boytsov. 2015. Decoding billions of integers per second with Python: analyzing text with the natural language toolkit. "O’Reilly Media, Inc."

[21] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy J. Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In REFLINLP.

[22] Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2012. Effect of Dynamic Pruning Safety on Learning to Rank Effectiveness. In Proceedings of the 35th
International ACM SIGIR Conference on Research and Development in Information Retrieval (Portland, Oregon, USA) (SIGIR ’12). Association for Computing Machinery, New York, NY, USA, 1051–1052.

[25] Joel Mackenzie, J. Shane Culpepper, Roi Blanco, Matt Crane, Charles L. A. Clarke, and Jimmy Lin. 2018. Query Driven Algorithm Selection in Early Stage Retrieval. In Proc. of the 11th ACM International Conference on Web Search and Data Mining. 396–404.

[26] Joel Mackenzie and Alistair Moffat. 2020. Examining the Additivity of Top-k Query Processing Innovations. In Proc. of the 29th ACM International Conference on Information and Knowledge Management. 1085–1094.

[27] Joel Mackenzie, Andrew Trotman, and Jimmy Lin. 2021. Wacky Weights in Learned Sparse Representations and the Revenge of Score-at-a-Time Query Evaluation. arXiv:2110.11540 [cs.IR]

[28] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning passage impacts for inverted indexes. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 1723–1727.

[29] Antonio Mallia, Giuseppe Ottaviano, Elia Porciani, Nicola Tonellotto, and Rossano Venturini. 2017. Faster BlockMax WAND with Variable-sized Blocks. In Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. 625–634.

[30] Antonio Mallia, Michal Siedlaczek, and Torsten Suel. 2019. An Experimental Study of Index Compression and DAAT Query Processing Methods. In Proc. of 41st European Conference on IR Research, ECIR’ 2019. 353–368.

[31] Antonio Mallia, Michal Siedlaczek, Mengyang Sun, and Torsten Suel. 2020. A Comparison of Top-k Threshold Estimation Techniques for Disjunctive Query Processing. In Proc. of the 29th ACM International Conference on Information and Knowledge Management. 2141–2144.

[32] Matthias Petri, J. Shane Culpepper, and Alistair Moffat. 2013. Exploring the Magic of WAND. In Proceedings of the 18th Australasian Document Computing Symposium (ADCS ’13). Association for Computing Machinery, New York, NY, USA, 58–65.

[33] Matthias Petri, Alistair Moffat, Joel Mackenzie, J. Shane Culpepper, and Daniel Beck. 2019. Accelerated Query Processing Via Similarity Score Prediction. In Proc. of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 485–494.

[34] Pyserini. 2021. https://github.com/castorini/pyserini.

[35] Cristian Rossi, Edleno S. de Moura, Andre L. Carvalho, and Altigran S. da Silva. 2013. Fast Document-at-a-time Query Processing using Two-tier Indexes. In Proc. of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. 183–192.

[36] Dongdong Shan, Shuai Ding, Jing He, Hongfei Yan, and Xiaoming Li. 2012. Optimized Top-k Processing with Global Page Scores on Block-Max Indexes. In Proc. of the 15th ACM International Conference on Web Search and Data Mining. 423–432.

[37] Jinjin Shao, Yifan Qiao, Shiyu Ji, and Tao Yang. 2021. Window Navigation with Adaptive Probing for Executing BlockMax WAND. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2323–2327.

[38] Trevor Strohman and W. Bruce Croft. 2007. Efficient document retrieval in main memory. In Proc. of the 30th ACM International Conference on Information and Knowledge Management. 63–72.

[39] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. 2013. Efficient and Effective Retrieval Using Selective Pruning. In Proc. of the Sixth ACM International Conference on Web Search and Data Mining (WSDM ’13). ACM, 63–72.

[40] 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), 319–500.

[41] Howard Turtle and James Flood. 1995. Query Evaluation: Strategies and Optimizations. Information Processing & Management 31, 6 (1995), 831–850.

[42] Erman Yafay and Ismail Sengor Altingovde. 2019. Caching Scores for Faster Query Processing with Dynamic Pruning in Search Engines. In Proc. of the 28th ACM International Conference on Information and Knowledge Management. 2457–2460.

[43] Yingrui Yang, Yifan Qiao, Jinjin Shao, Mayuresh Vivekanand Anand, Xifeng Yan, and Tao Yang. 2021. Composite Re-Ranking for Efficient Document Search with BERT. WSDM’22 and ArXiv abs/2103.06499 (2021).

[44] Hamed Zamani, Mostafa Dehghani, William Bruce Croft, Erik G. Learned-Miller, and J. Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018).