Representation Sparsification with Hybrid Thresholding for Fast SPLADE-based Document Retrieval

Yifan Qiao
Department of Computer Science, University of California
Santa Barbara, California, USA

Yingrui Yang
Department of Computer Science, University of California
Santa Barbara, California, USA

Shanxiu He
Department of Computer Science, University of California
Santa Barbara, California, USA

Tao Yang
Department of Computer Science, University of California
Santa Barbara, California, USA

ABSTRACT

Learned sparse document representations using a transformer-based neural model have been found to be attractive in both relevance effectiveness and time efficiency. This paper describes a representation sparsification scheme based on hard and soft thresholding with an inverted index approximation for faster SPLADE-based document retrieval. It provides analytical and experimetal results on the impact of this learnable hybrid thresholding scheme.

CCS CONCEPTS

• **Information systems → Retrieval efficiency**.

KEYWORDS

Learned sparse representations, top-k retrieval, index pruning.

ACM Reference Format:
Yifan Qiao, Yingrui Yang, Shanxiu He, and Tao Yang. 2023. Representation Sparsification with Hybrid Thresholding for Fast SPLADE-based Document Retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3592051

1 INTRODUCTION

Recently learned sparse retrieval techniques [5–8, 10, 20, 23, 37] have become attractive because such a representation can deliver a strong relevance by leveraging transformer-based models to expand document tokens with learned weights and can take an advantage of traditional inverted index based retrieval techniques [24, 25]. Its query processing is cheaper than a dense representation which requires GPU support (e.g. [31, 32, 35]) even with efficiency optimization through approximate nearest neighbor search [14, 34, 38].

This paper focuses on the SPLADE family of sparse representations [6–8] because it can deliver a high MRR@10 score for MS MARCO passage ranking [4] and a strong zero-shot performance for the BEIR datasets [33], which are well-recognized IR benchmarks. The sparsification optimization in SPLADE has used L1 and FLOPS regularization to minimize non-zero weights during model learning, and our objective is to exploit additional opportunities to further increase the sparsity of inverted indices produced by SPLADE. Earlier static inverted index pruning research [1–3] for a lexical model has shown the usefulness of trimming a term posting list or a document by a limit. Yang et al. [36] conduct top token masking by limiting the top activated weight count uniformly per document and gradually reduce this weight count limit to a targeted constant when training SPLADE. Motivated by these studies [1–3, 36] and since they have not addressed learnability of a pruning limit through relevance-driven training, this paper exploits a learnable thresholding architecture to filter out unimportant neural weights produced by the SPLADE model through joint training.

The contribution of this paper is a learnable hybrid hard and soft thresholding scheme with an inverted index approximation to increase the sparsity of SPLADE-based document and query feature vectors for faster retrieval. In addition to experimental validation with MS MARCO and BEIR datasets, we provide an analysis on the impact of hybrid thresholding with joint training on index approximation errors and training update effectiveness.

2 BACKGROUND

For a query q and a document d, after expansion and encoding, they can be represented by vector $\vec{w}(q)$ and $\vec{w}(d)$ with length $|V|$, where V is the vocabulary set. The rank score of q and d is computed as $R(q,d) = \vec{w}(q) \cdot \vec{w}(d) = \sum_{i=1}^{|V|} w_i^q \times w_i^d$. For sparse vectors with many zeros, retrieval can utilize a data structure called inverted index during online inference for fast score computation [24, 25]. The SPLADE model uses the BERT token space to predict the feature vector $\vec{w}$. In its latest SPLADE++ model, it first calculates the importance of i-th input token in d for each j in V: $w_{ij}(\Theta) = \text{Transform}(\vec{h_i})^T \vec{E_j} + b_j$, where $\vec{h_i}$ is the BERT embedding of i-th token in d, $\vec{E_j}$ is the BERT input embedding for j-th token. Transform() is a linear layer with GeLU activation and LayerNorm. The weights in this linear layer, $\vec{E_j}$, and b_j are the SPLADE parameters updated during training and we call them set Θ. Then the j-th entry w_j of document d (or a query) is max-pooled as $w_j(\Theta) = \max_{i \in d}\{\log(1 + \text{ReLU}(w_{ij}(\Theta)))\}$. Notice that $w_j \geq 0$.

The loss function of SPLADE models [6–8] contains a per-query ranking loss L_R and sparsity regularization. The ranking loss has evolved from a log likelihood based function for maximizing positive document probability to margin MSE for knowledge distillation. This paper uses the loss of SPLADE with a combination that delivers the best result in our training process. L_R is the ranking loss with margin MSE for knowledge distillation [12]. The document token regularization L_D is computed on the training documents in

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3592051

each batch based on FLOPS regularization. The query token regularization L_Q is based on L1 norm. Let B be a set of training queries with N documents involved in a batch. $L_Q = \sum_{j\in V} \frac{1}{|B|} \sum_{q\in B} w_j^q$; $L_D = \sum_{j\in V} (\frac{1}{N} \sum_{d=1}^{N} w_j^d)^2$.

Related work. Other than SPLADE, sparse retrieval studies include SNRM [37], DeepCT [5], DeepImpact [23], and uniCOIL [10, 20]. The sparsity of a neural network is studied in the deep learning community. Soft thresholding in [16] adopts a learnable threshold with function $S(x,t) = ReLU(x-t)$ to make parameter x zero under threshold t. A hard thresholding function $H(x,t) = x$ when $x \geq t$ otherwise 0. Approximate hard thresholding [28] uses a Gauss error function to approximate $H(x,t)$ with smooth gradients. Dynamic sparse training [21] finds a dynamic threshold with marked layers. These works including the recent ones [9] are targeted for sparsification of parameter edges in a deep neural network. In our context, a token weight w_j is an output node in a network. The sparsification of output nodes is addressed in activation map compression [11] using ReLU as soft thresholding together with L1 regularization. The work of [15] further boosts sparsity with the Hoyer regularization and a variant of ReLU. The above techniques have not been investigated in the context of sparse retrieval, and the impact of thresholding on relevance and query processing time with inverted indices, requires new design considerations and model structuring for document retrieval, even the previous work can be leveraged.

3 HYBRID THRESHOLDING (HT)

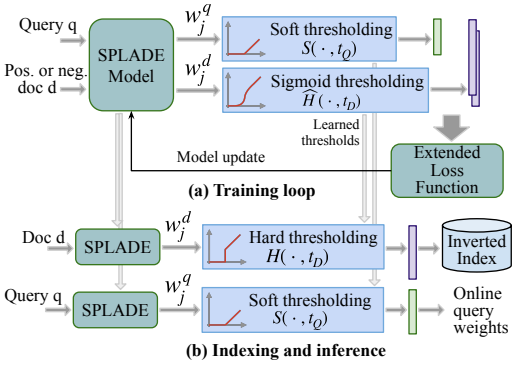

Figure 1: Hybrid thresholding with an index approximation

Design considerations. To zero out a token weight below a learnable threshold, there are two options: soft thresholding [16], and approximate hard thresholding [28]. For query token weights, we find that soft thresholding does not affect relevance significantly. For document token weights, our study finds that compared to soft thresholding, hard thresholding can retain relevance better since it does not change token weights when exceeding a threshold. Since the subgradient for hard thresholding with respect to a threshold is always 0, an approximation needs to be carried out for training. For search index generation, an inverted index produced with the same approximate hard thresholding as training keeps many unnecessary non-zero document token weights, slowing down retrieval significantly. Thus we directly apply hard thresholding with a threshold learned from training, as shown in Figure 1. There is a gap between trained document token weights and actual weights used in our inverted index generation and online inference, and we intend to minimize this gap (called an index approximation error).

Thus our design takes a hybrid approach that applies soft thresholding to query token weights during training and inference and applies approximate hard thresholding to document token weights during training while using hard thresholding for documents during index generation. For approximate hard thresholding, we propose to use a logistic sigmoid-based function instead of a Gauss error function [28]. This sigmoid thresholding simplifies our analysis of the impact of its hyperparameter choice to index approximation errors, and to training stability.

3.1 Trainable and approximate thresholding

Training computes threshold parameters t_D, and t_Q for documents and queries, respectively. From the output of the SPLADE model, every token weight of a query is replaced with $S(w_j^q, t_Q)$, which is $ReLU(w_j^q - t_Q)$, and every document token weight is replaced with $\widehat{H}(w_j^q, t_D)$ before their dot product is computed during training as shown in Figure 1(a). Sigmoid thresholding $\widehat{H}$ is defined as:

$$\widehat{\mathrm{H}}(w_j^d, t_D) = w_j^d \sigma(K(w_j^d - t_D)) \text{ where } \sigma(x) = \frac{1}{1+e^{-x}}. \quad (1)$$

Here K is a hyperparameter to control the slope steepness of step approximation that jumps from 0 to 1 when exceeding a threshold.

The indexing process uses hard thresholding to replace all document weights that are below threshold t_D as 0 as depicted in Figure 1(b). The above post processing introduces an index approximation error $E = |\widehat{H}(w_j^d, t_D) - H(w_j^d, t_D)|$. We derive its upper bound as follows. Notice that $w_j \geq 0$, and for any $x \geq 0$, $1 + x \leq e^x$.

$$E = w_j^d \sigma(K(w_j^d - t_D)) = \frac{w_j^d}{1 + e^{K(t_D - w_j^d)}} \leq \frac{w_j^d}{2 + K(t_D - w_j^d)}.$$

When $w_j^d \geq t_D$, we can derive that

$$E = w_j^d(1 - \sigma(K(w_j^d - t_D))) = w_j^d \sigma(K(t_D - w_j^d))) \leq \frac{w_j^d}{2 + K(w_j^d - t_D)}.$$

Let σ^- denote $\sigma(K(w_i^d - t_D))$. $0 < \sigma^- < 1$. In both of the above cases, the error upper bound is minimized when K is large. This is consistent with the fact that error E is monotonically decreasing as K increases because $\frac{\partial E}{\partial K} = -w_j^d \sigma^-(1-\sigma^-)|w_j^d - t_D| \leq 0$. When $|w_j^d - t_D|$ is big, the error is negligible and when $|w_j^d - t_D|$ is small, the error could become big with a small K value. But as shown later, an excessively large K value could cause a big parameter update during a training step, affecting joint training stability.

Let $Dlen$ and $Qlen$ be the non-zero token weight count of document d and query q, respectively. For our hybrid thresholding, $Dlen = \sum_j \mathbf{1}_{w_j^d \geq t_D}$, $Qlen = \sum_j \mathbf{1}_{w_j^q \geq t_Q}$. Here $\mathbf{1}_{x \geq y}$ is an indicator function as 1 if $x \geq y$ otherwise 0. When increasing t_D and t_Q, $Dlen$ and $Qlen$ decrease. Thus for a batch of training queries B, the original SPLADE loss is extended as: $L = (\frac{1}{|B|} \sum_{q\in B} L_R) + \lambda_Q L_Q + \lambda_D L_D + \lambda_T L_T$. The extra item added is $L_T = \log(1 + e^{-t_D}) + \log(1 + e^{-t_Q})$. We retain the original L_Q and L_D expressions because as w_j^q or w_j^d decreases, more weights can quickly be zeroed out.

3.2 Threshold and token weight updating

We study the change of t_D, t_Q, w_j^d, and w_j^q after each training step with a mini-batch gradient descent update. The analysis below uses the first-order Taylor polynomial approximation and follows the fact that sigmoid thresholding $\widehat{H}$ and soft thresholding function S are used independently for a query and a document in the loss function. Symbol α is the learning rate. Let "$d \triangleleft q$" mean d is a positive or negative document of query q.

$$\Delta t_D = t_D^{new} - t_D^{old} = -\alpha \frac{\partial L}{\partial t_D} = -\alpha \left(\frac{1}{|B|} \sum_{q \in B} \frac{\partial L_R}{\partial \widehat{H}} \frac{\partial \widehat{H}}{\partial t_D} + \lambda_T \frac{\partial L_T}{\partial t_D} \right)$$

$$= \alpha \left(\frac{1}{|B|} \sum_{q \in B} \left(K \frac{\partial L_R}{\partial \widehat{H}} \sum_{d \triangleleft q} \sum_i w_i^d (1-\sigma)^- \sigma^- \right) + \lambda_T \frac{e^{-t_D}}{1+e^{-t_D}} \right).$$

$$\Delta t_Q = t_Q^{new} - t_Q^{old} = -\alpha \frac{\partial L}{\partial t_Q} = -\alpha \left(\frac{1}{|B|} \sum_{q \in B} \frac{\partial L_R}{\partial S} \frac{\partial S}{\partial t_Q} + \lambda_T \frac{\partial L_T}{\partial t_Q} \right)$$

$$= \alpha \left(\frac{1}{|B|} \sum_{q \in B} \left(\frac{\partial L_R}{\partial S} \sum_i \mathbf{1}_{w_i^q \geq t_Q} \right) + \lambda_T \frac{e^{-t_Q}}{1+e^{-t_Q}} \right).$$

$$\Delta w_j^d = w_j^{d,new} - w_j^{d,old} \approx \sum_{\theta \in \Theta} \frac{\partial w_j^d}{\partial \theta} \Delta\theta = -\sum_{\theta \in \Theta} \frac{\partial w_j^d}{\partial \theta} \alpha \frac{\partial L}{\partial \theta}$$

$$= -\alpha \sum_{\theta \in \Theta} \frac{\partial w_j^d}{\partial \theta} \left(\frac{1}{|B|} \sum_{q \in B} \left(\frac{\partial L_R}{\partial \widehat{H}} \left(\sum_{d \triangleleft q} \sum_i \frac{\partial \widehat{H}}{\partial w_i^d} \frac{\partial w_i^d}{\partial \theta} \right) + \frac{\partial L_R}{\partial S} \left(\sum_i \frac{\partial S}{\partial w_i^q} \frac{\partial w_i^q}{\partial \theta} \right) \right) + \lambda_D \frac{\partial L_D}{\partial \theta} + \lambda_Q \frac{\partial L_Q}{\partial \theta} \right).$$

Notice that $\frac{\partial \widehat{H}}{\partial w_i^d} = \sigma^- + K w_i^d \sigma^-(1-\sigma^-)$. The above results indicate:

- A significant number of terms in Δt_D and Δw_j^d involve linear coefficient K. This is verifiably true also for Δw_j^q. Although a large K value can minimize the index approximation error $|\widehat{H}(w_j^d, t_D) - H(w_j^d, t_D)|$, it can cause an aggressive change of token weights and thresholds at a training iteration, making training overshoot and miss the global optimum. Thus K cannot be too large, and our evaluation further studies this.
- If $\frac{\partial L_R}{\partial \widehat{H}} \geq 0$, $\Delta t_D \geq 0$, and the document threshold increases, decreasing *Dlen*. Otherwise document token threshold may decrease after a parameter update step during training, and the degree of decreasing is reduced by a positive value $\frac{e^{-t_D}}{1+e^{-t_D}}$. Based on the sign of $\frac{\partial L_R}{\partial S}$, we can draw a similar conclusion on Δt_Q.

4 EVALUATION

Our evaluation uses MS MARCO passages [4] and BEIR datasets [33]. MS MARCO has 8.8M passages while BEIR has 13 different datasets of varying sizes up-to 5.4M. As a common practice, we report the relevance in terms of mean reciprocal rank MRR@10 for the MS MARCO passage Dev query set with 6980 queries, and the normalized discounted cumulative gain nDCG@10 [13] for its DL'19 and DL'20 sets, and also for BEIR. For retrieval with a SPLADE inverted index, we report the mean response time (MRT) and 99th percentile time (P_{99}) in milliseconds. The query encoding time is not included. For the SPLADE model, we warm up it following [7, 17], and train it with $\lambda_Q = 0.01$ and $\lambda_D = 0.008$, and hybrid thresholding. We use the PISA [26] search system to index documents and search queries using SIMD-BP128 compression [18] and MaxScore retrieval [24, 27]. Our evaluation runs as a single thread on a Linux CPU-only server with Intel i5-8259U 2.3GHz and 32GB memory. Similar retrieval latency results are observed on a 2.3GHz AMD EPYC 7742 processor. The checkpoints and related code will be released in https://github.com/Qiaoyf96/HT.

Table 1: Overall results on MS MARCO passages

Methods	MRR Dev	MRT(P_{99}) k = 10	MRT(P_{99}) k = 1000	nDCG DL'19	nDCG DL'20	Dlen
SPLADE	**0.3966**	48.3(228)	127(408)	**0.7398**	**0.7340**	351
/DT [28]	0.3922	102(457)	262(786)	0.7392	0.7319	444
/Top305 [36]	0.3962	42.4(202)	114(369)	0.7353	0.7288	277
/Top100 [36]	0.3908	21.8(106)	62.5(196)	0.7192	0.7119	99
/DCP50% [2]	0.3958	30.0(145)	83.9(271)	0.7385	0.7321	175
/DCP40% [2]	0.3933	25.9(124)	73.3(235)	0.7335	0.7280	140
/DCP30% [2]	0.3912	21.6(101)	61.8(193)	0.7287	0.7217	105
/Cut0.5	0.3924	21.9(104)	62.6(195)	0.7296	0.7212	144
/Cut0.8	0.3885	15.6(70.4)	43.8(128)	0.7207	0.7118	112
/HT_1	0.3955	22.8(108)	62.3(195)	0.7322	0.7210	140
/HT_3	0.3942	14.2(67.2)	40.6(123)	0.7327	0.7228	106
/HT_1-2GTI [30]	0.3959	10.0(49.1)	27.6(92.2)	0.7330	0.7210	140
/HT_3-2GTI [30]	0.3942	**6.9(33.9)**	**19.3(62.1)**	0.7320	0.7228	106

Overall results with MS MARCO. Table 1 is a comparison with the baselines on MS MARCO passage Dev set, DL'19, and DL'20. It lists the average *Dlen* value, and top-k retrieval time with depth $k = 10$ and 1000. Row 3 is for original SPLADE trained by ourselves with an MRR number higher than 0.38 reported in [7, 17]. Rows 12 and 13 list the result of our hybrid thresholding marked as HT_{λ_T} and $K = 25$. With $\lambda_T = 1$, SPLADE/HT_1 converges to a point where $t_Q = 0.4$ and $t_D = 0.5$, which is about 2x faster in retrieval. HT_3 with $\lambda_T = 3$ converges at $t_Q = 0.7$ and $t_D = 0.8$, resulting 3.1x speedup than SPLADE while having a slightly lower MRR@10 0.3942. No statistically significant degradation in relevance has been observed at the 95% confidence level for both HT_1 and HT_3. The inverted index size reduces from 6.4GB for original SPLADE to 2.8GB and 2.2GB for HT_1 and HT_3 respectively. When applying two-level guided traversal 2GTI [30] with its fast configuration, Rows 14 and 15 show a further latency reduction to 6.9ms or 19.3ms.

We discuss other baselines listed in this table. Row 4 named DT uses the thresholding scheme from [28]. Its training does not converge with its loss function, and its retrieval is much slower. Rows 5 and 6 follow joint training of top-k masking [36] with the top 305 tokens as suggested in [36] and with the top 100 tokens. Rows 7, 8 and 9 marked with DCPx follow document centric pruning [2] that keeps x of top tokens per document where x=50%, 40%, and 30%. We did not list term centric pruning [1, 3] because [2] shows DCP is slightly better in relevance under the same latency constraint. Rows 10 and 11 with "/Cut0.5" and "/Cut0.8" apply a hard threshold with 0.5 and 0.8 in the output of original SPLADE without joint training. The index pruning options without learning from Rows 5 to 11 can either reduce the latency to the same level as HT, but their relevance score is visibly lower; or have a relevance similar to HT but with much slower latency. This illustrates the advantage of learned hybrid thresholding with joint training.

Table 2 lists the zero-shot performance of HT when $k = 1000$ by applying the SPLADE/HT model learned from MS MARCO to the

Table 2: Zero-shot performance on BEIR datasets

	SPLADE		SPLADE/HT_1		SPLADE/HT_3	
Dataset	**nDCG**	**MRT**	**nDCG**	**MRT**	**nDCG**	**MRT**
DBPedia	0.430	135	**0.435**	64.2	0.426	32.3
FiQA	**0.354**	6.5	0.345	4.0	0.336	3.2
NQ	**0.547**	81.8	0.545	45.9	0.539	28.6
HotpotQA	0.678	481	**0.680**	265	0.678	140
NFCorpus	0.351	0.5	**0.352**	0.3	0.346	0.2
T-COVID	0.719	16.0	**0.730**	10.1	0.695	7.5
Touche-2020	0.307	15.0	0.306	9.3	**0.313**	4.5
ArguAna	0.440	20.8	0.463	7.8	**0.500**	4.1
C-FEVER	**0.234**	1375	0.219	681	0.213	332
FEVER	**0.781**	1584	0.778	559	0.764	264
Quora	**0.806**	17.5	0.776	9.2	0.792	4.5
SCIDOCS	0.151	6.9	**0.155**	3.0	0.151	2.0
SciFact	0.676	5.7	**0.681**	2.4	0.672	1.4
Average	**0.498**	-	0.497	2.0x	0.494	3.6x

BEIR datasets without any additional training. HT_1 has a similar nDCG@10 score as SPLADE without HT, while having a 2x MRT speedup on average. HT_3 is even faster with 3.6x speedup, and its nDCG@10 drops in some degree to 0.494.

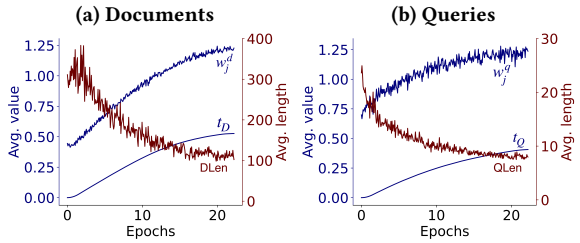

Figure 2: Weight/threshold/sparsity changes during training

Figure 2 depicts the average values of w_j^d, t_D, and *Dlen* on the left and w_j^q, t_Q, and *Qlen* on the right during MS MARCO training under HT_1. x-axis is the training epoch number. It shows that *Dlen* and *Qlen* decrease while t_D and t_Q increase as training makes a progress and SPLADE/HT_1 converges after about 20 epochs.

Design options. Table 3 lists performance under 4 thresholding combinations from Row 3 to Row 7. $S[x]$ means soft thresholding function $S()$ is applied to x for both training and indexing where x can be documents (D) or queries (Q). $\widehat{H}[x]$ means sigmoid thresholding $\widehat{H}$ is applied in both training and indexing. $\widehat{H}H[x]$ means $\widehat{H}$ is applied in training and H is applied in indexing. $\phi[x]$ means no thresholding is applied to x during training and indexing. When thresholding is not applied to queries, $\widehat{H}H[D]$ is 1.3x faster than $S[D]$ when $k = 10$ and $k = 1000$ while their relevance scores are similar. Shifting of document weight distribution by soft thresholding significantly affects retrieval time. Rows 6 and 7 fix $\widehat{H}H[D]$ setting, and show that soft thresholding is more effective in relevance than hard thresholding for query tokens. Shifting of query weight distribution has less effect on latency while gaining more relevance through model consistency between training and indexing.

Hyperparameter K in sigmoid thresholding $\widehat{H}$. Table 3 compares $\widehat{H}H[D]$ with $\widehat{H}[D]$ when varying K from Row 8 to Row 14. In these cases, training always uses $\widehat{H}$ while indexing uses $\widehat{H}$ or H. When K is small as 2.5, applying $\widehat{H}$ to both training and indexing yields good relevance, but retrieval is about 1.8x slower

Table 3: Impact of design options. MS MARCO passages.

HT Config. $\lambda_T = 1$	**MRR**	**MRT(P_{99})** $k = 10$	**MRT(P_{99})** $k = 1000$	**Qlen**	**Dlen**
Soft vs. hard thresholding in 4 combinations. Fix $K = 25$.					
$\phi[Q], S[D]$	0.3941	31.7(157)	91.5(315)	14.3	145
$\phi[Q], \widehat{H}H[D]$	0.3942	24.1(111)	70.7(219)	13.5	142
$S[Q], \widehat{H}H[D]$	**0.3955**	**22.8(108)**	**62.3**(195)	11.3	140
$\widehat{H}H[Q], \widehat{H}H[D]$	0.3904	24.9(106)	62.6(**182**)	9.0	142
Vary K. $\widehat{H}[D]$ vs. $\widehat{H}H[D]$. Fix $S[Q]$.					
$\widehat{H}H[D]$, $K = 2.5$	0.3947	22.8(110)	62.6(199)	11.5	149
$\widehat{H}[D]$, $K = 2.5$	**0.3963**	41.4(198)	112(358)	11.5	421
$\widehat{H}H[D]$, $K = 25$	0.3955	22.8(108)	62.3(195)	11.3	140
$\widehat{H}[D]$, $K = 25$	0.3961	28.7(136)	76.9(239)	11.3	208
$\widehat{H}H[D]$, $K = 250$	0.3946	**21.9(102)**	**60.5(189)**	11.2	135
$\widehat{H}[D]$, $K = 250$	0.3947	23.1(112)	63.9(203)	11.2	159
Usefulness of L_Q and L_D. Fix $S[Q]$, $\widehat{H}H[D]$, and $K = 25$.					
w/o L_Q	0.3956	56.2(245)	166(502)	20.1	138
w/o L_Q, L_D	0.3954	99.4(434)	254(772)	25.9	421

because much more non-zero weights are kept in the index. When K becomes large as 250, training does not converge to the global optimum due to large update sizes, resulting in an MRR score lower than K=25 even with no index approximation. $K = 25$ has a reasonable MRR while $\widehat{H}H[D]$ is up-to 26% faster than $\widehat{H}[D]$.

Retaining L_Q and L_D. Last three rows of Table 3 shows that the query length is higher when L_Q is removed from the loss function, and documents get longer when L_D is removed further. The result means L_Q and L_D are useful in sparsity control together with L_T.

5 CONCLUDING REMARKS

Our evaluation shows that learnable hybrid thresholding with index approximation can effectively increase the sparsity of inverted indices with 2-3x faster retrieval and competitive or slightly degraded relevance (0.28% - 0.6% MRR@10 drop). Its trainability allows relevance and sparsity guided threshold learning and it can outperform index pruning without such an optimization. Our scheme retains a non-uniform number of non-zero token weights per vector based on a trainable weight and threshold difference for flexibility in relevance optimization. Our analysis shows that hyperparameter K in sigmoid thresholding needs to be chosen judiciously for a small index approximation error without hurting training stability.

If a small relevance tradeoff is allowed, more retrieval time reduction is possible when applying other related orthogonal efficiency optimization techniques [17, 19, 22, 24, 29, 30]. Applying hybrid thresholding HT_3 to a checkpoint of a recent efficiency-driven SPLADE model [17] with 0.3799 MRR@10 on the MS MARCO passage Dev set, decreases the response time from 36.6ms to 21.7ms (1.7x faster) when k=1000 while having 0.3868 MRR@10. This latency can be further reduced to 14.2ms with the same MRR@10 number (0.3868) when 2GTI [30] is applied to the above index.

A future study is to investigate the use of the proposed hybrid thresholding scheme for other learned sparse models [10, 20, 23].

Acknowledgments. We thank Wentai Xie and anonymous referees for their valuable comments and/or help. This work is supported in part by NSF IIS-2225942 and has used computing resource of NSF's ACCESS program. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Roi Blanco and Alvaro Barreiro. 2007. Boosting Static Pruning of Inverted Files. In *Proc. of SIGIR* (Amsterdam, The Netherlands) *(SIGIR '07)*. Association for Computing Machinery, New York, NY, USA, 777–778. https://doi.org/10.1145/1277741.1277904

[2] 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* (Arlington, Virginia, USA) *(CIKM '06)*. Association for Computing Machinery, New York, NY, USA, 182–189. https://doi.org/10.1145/1183614.1183644

[3] 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 *Proc. of SIGIR* (New Orleans, Louisiana, USA) *(SIGIR '01)*. Association for Computing Machinery, New York, NY, USA, 43–50. https://doi.org/10.1145/383952.383958

[4] 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).

[5] 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).

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

[7] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2022. From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022).

[8] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 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).

[9] Elias Frantar and Dan Alistarh. 2022. SPDY: Accurate Pruning with Speedup Guarantees. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 6726–6743. https://proceedings.mlr.press/v162/frantar22a.html

[10] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. *NAACL* (2021).

[11] Georgios Georgiadis. 2019. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7085–7095.

[12] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *ArXiv* abs/2010.02666 (2020).

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Trans. on Big Data* 7, 3 (2019), 535–547.

[15] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. 2020. Inducing and Exploiting Activation Sparsity for Fast Inference on Deep Neural Networks. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 5533–5543. https://proceedings.mlr.press/v119/kurtz20a.html

[16] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*. PMLR, 5544–5555.

[17] Carlos Lassance and Stéphane Clinchant. 2022. An efficiency study for SPLADE models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2220–2226.

[18] Daniel Lemire and Leonid Boytsov. 2015. Decoding billions of integers per second through vectorization. *Softw. Pract. Exp.* 45, 1 (2015), 1–29.

[19] Jimmy Lin and Andrew Trotman. 2015. Anytime Ranking for Impact-Ordered Indexes. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval* (Northampton, Massachusetts, USA) *(ICTIR '15)*. Association for Computing Machinery, New York, NY, USA, 301–304. https://doi.org/10.1145/2808194.2809477

[20] Jimmy J. Lin and Xueguang Ma. 2021. A Few Brief Notes on DeepImpact, COIL, and a Conceptual Framework for Information Retrieval Techniques. *ArXiv* abs/2106.14807 (2021).

[21] Junjie LIU, Zhe XU, Runbin SHI, Ray C. C. Cheung, and Hayden K.H. So. 2020. Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJlbGJrtDB

[22] Joel Mackenzie, Matthias Petri, and Alistair Moffat. 2021. Anytime Ranking on Document-Ordered Indexes. *ACM Trans. Inf. Syst.* 40, 1, Article 13 (sep 2021), 32 pages.

[23] Antonio Mallia, O. Khattab, Nicola Tonellotto, and Torsten Suel. 2021. Learning Passage Impacts for Inverted Indexes. *SIGIR* (2021).

[24] Antonio Mallia, Joel Mackenzie, Torsten Suel, and Nicola Tonellotto. 2022. Faster Learned Sparse Retrieval with Guided Traversal. In *Proceedings of the 45th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022)*. 1901–1905.

[25] 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.

[26] Antonio Mallia, Michal Siedlaczek, Joel Mackenzie, and Torsten Suel. 2019. PISA: Performant indexes and search for academia. *Proceedings of the Open-Source IR Replicability Challenge* (2019).

[27] 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.

[28] Jin-Woo Park and Jong-Seok Lee. 2020. Dynamic Thresholding for Learning Sparse Neural Networks. In *ECAI 2020*. IOS Press, 1403–1410.

[29] Yifan Qiao, Yingrui Yang, Haixin Lin, Tianbo Xiong, Xiyue Wang, and Tao Yang. 2022. Dual Skipping Guidance for Document Retrieval with Learned Sparse Representations. *ArXiv* abs/2204.11154 (April 2022).

[30] Yifan Qiao, Yingrui Yang, Haixin Lin, and Tao Yang. 2023. Optimizing Guided Traversal for Fast Learned Sparse Retrieval. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*. ACM, Austin, TX, USA.

[31] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2825–2835.

[32] Keshav Santhanam, O. Khattab, Jon Saad-Falcon, Christopher Potts, and Matei A. Zaharia. 2021. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *ArXiv* abs/2112.01488 (16 Dec. 2021).

[33] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=wCu6T5xFjeJ

[34] Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Defu Lian, Yeyun Gong, Qi Chen, Fan Yang, Hao Sun, Yingxia Shao, Denvy Deng, Qi Zhang, and Xing Xie. 2022. Distill-VQ: Learning Retrieval Oriented Vector Quantization By Distilling Knowledge from Dense Embeddings. *Proc. of SIGIR* (2022).

[35] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*. https://openreview.net/forum?id=zeFrfgyZln

[36] Jheng-Hong Yang, Xueguang Ma, and Jimmy Lin. 2021. Sparsifying Sparse Representations for Passage Retrieval by Top-k Masking. *CoRR* abs/2112.09628 (2021). arXiv:2112.09628 https://arxiv.org/abs/2112.09628

[37] 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).

[38] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *Proc. of Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22)*. ACM, New York, NY, USA, 1328–1336.