Token Pruning Optimization for Efficient Multi-Vector Dense Retrieval

Shanxiu He[1 [0009−0008−8581−6733]], Mutasem Al-Darabsah[2 [0009−0003−3986−6311]], Suraj Nair[2 [0000−0003−2283−7672]], Jonathan May[2,3 [0000−0002−5284−477X]], Tarun Agarwal[2 [0009−0004−5682−8234]], Tao Yang[1 [0000−0003−1902−3387]], and Choon Hui Teo[2 [0000−0002−5724−9478]]

[1] University of California, Santa Barbara, USA
{shanxiuhe,tyang}@cs.ucsb.edu
[2] Amazon, Palo Alto, USA
{mutasema, srjnair, jnatmay, tagar, choonhui}@amazon.com
[3] University of Southern California, Los Angeles, USA
jonmay@isi.edu

Abstract. Multi-vector dense retrieval with ColBERT has been shown to be effective in striking a good relevance and efficiency tradeoff for both in-domain and out-of-domain datasets through late interaction between queries and documents. However, the efficiency of ColBERT for a large-scale retrieval dataset is still constrained by its large memory footprint, as one embedding is stored per token; thus, previous work has studied static pruning of less significant tokens to enhance efficiency. To improve the adaptivity of prior work in zero-shot retrieval settings, this paper proposes a neural classification method that learns pruning decisions with Gumbel-Softmax, and provides an extension to adjust pruning decisions and meet memory space reduction requirements. We evaluate the effectiveness of our proposed method against several baseline approaches on out-of-domain datasets LoTTE and BEIR, and the in-domain MS MARCO passage dataset.

Keywords: Multi-vector neural representations · Late-interaction dense retrieval · Pruning · Space and time efficiency

1 Introduction

A transformer-based cross-encoder for document ranking delivers impressive performance but comes with extremely high computational costs during inference. To reduce complexity, many dense retrieval methods adopt a single-vector paradigm to represent queries and documents. However, deploying such single-vector models for large search datasets with diverse content domains and applications, such as product search involving a large number of entities, is challenging. As pointed out in the previous work [16, 25, 26], the single-vector representation paradigms have limited expressive power when handling out-of-domain datasets

with scarce training data (including zero-shot retrieval) and in answering entity-centric questions. ColBERT [12] with a multi-vector representation can be viewed as a middle-ground between the single-vector bi-encoder and the cross-encoder design. Previous studies demonstrate that, compared to sparse representations like SPLADE [7, 8], ColBERTv2 can achieve comparable performance on the MS MARCO and BEIR datasets [23, 28], while also offering a relevance advantage when searching an out-of-domain data collection called LoTTE [23]. As a result, multi-vector dense representations can be desirable in certain search applications, especially on GPU-rich platforms where sparse retrieval cannot fully leverage GPU resources.

The expressiveness advantage of ColBERT over single-vector representations comes at a significant cost, namely, an order of magnitude increase in space and computational complexity. Reducing space complexity is important as the memory size of GPUs is much more limited than CPUs and maintaining high memory bandwidth becomes challenging as GPU memory capacity expands. Several efficiency optimization approaches have been developed recently for ColBERT, including offline techniques for space reduction such as quantization to compress the vector size [23], and token pruning [1, 14]. Approaches of online inference acceleration include DESSERT [6], EMVB [20], CITADEL [16], and XTR [15].

Our paper focuses on token pruning optimization to minimize the number of tokens representing each document, thereby reducing space usage and accelerating inference time. Existing techniques like Top-k token pruning [14] face limitations due to the static nature of parameter k, which is not learned, making them less effective for out-of-domain search. Similarly, IDF-based pruning [1], while effective in some cases, can be unsafe for certain domains as removing frequently occurring tokens might inadvertently discard critical information. Increasing the IDF method parameter often leads to a significant relevance drop under tight space budgets. These challenges highlight the need for learnable context-aware pruning to improve adaptivity.

This paper develops a context-aware neural module that decides the pruning of tokens for each document in ColBERT-like dense retrieval. Our method, **LeapMV** (**Lea**rnable **P**runing for **M**ulti-**V**ector representations), is trained to minimize the multi-vector dense retrieval loss for one domain and is applicable to zero-shot retrieval in searching out-of-domain datasets. LeapMV can be extended to adjust its pruning aggressiveness to meet different space reduction requirements. Our evaluation shows that LeapMV improves the adaptivity of the previously proposed static pruning techniques in meeting different space reduction targets, which results in latency saving proportionally. Under the same relevance accuracy budget for LoTTE, LeapMV consumes up to 3.3× less space and is up-to 2.8× faster than the top-k token pruning and IDF methods.

2 Background and Design Considerations

The multi-vector representation in ColBERT encodes the tokens in each document (and a query) using embeddings and its index is built based on a set of

token embedding vectors for each document. During inference time, ColBERT calculates the similarity between a document d and the given query q via the following "MaxSim" operations,

$$S_{q,d} = \sum_{i=1}^{N} \max_{j=1}^{M} Q_i \cdot D_j^T \tag{1}$$

where Q is the encoded token embedding set for a given query and D the token embedding sequence for each document d. N and M are the token sequence length for a query and a document respectively.

The required ColBERT online memory size for hosting the compressed multi-vector document representation is approximately proportional to the product of the average document length, the token embedding dimension, and the number of documents. This can easily flood out memory space for hosting a large search data collection with a long language-model embedding length. Furthermore, the online inference time complexity is proportional to the average number of tokens per document. Thus reducing the number of tokens per document can lead to near-linear decreases in the memory usage and search inference time.

Top-k token pruning. Static token pruning in [14] keeps top-k tokens per document, and several heuristics have been proposed to select such k items. Although the ColBERT model is optimized through training based on the given k hyperparameter, the trained ColBERT model based on one domain (e.g. MS MARCO) with a fixed k value can lose its advantage when this model is directly applied in a zero-shot manner to other search domains which have different document characteristics including lengths and IDF distributions.

IDF-based token pruning. The study in [1] compares several IDF-based static pruning methods for ColBERT with MS MARCO and finds the best method is called IDF uniform which removes top-τ BERT tokens (including stopwords) globally in all documents. Uniformly pruning popular words in all documents can inadvertently remove meaningful tokens in some contexts. For example, "the who" is the name of a well-known rock band, while "who" can stand for the World Health Organization–both of which could be pruned by the above IDF uniform method. When a larger space reduction is required, increasing the IDF filtering parameter can incur a significant relevance loss as demonstrated in our evaluation. Thus we seek a learned neural classifier for context-aware token pruning, applicable for zero-shot retrieval.

Text chunking. We have also considered using text chunking, which originated in NLP research [22]. Chunking divides text into syntactically related, non-overlapping groups of words and reduces the number of representations per document by considering each chunk as a merged token. We have applied text chunking such as sentence chunking and word chunking for ColBERT and have found significant relevance degradations of such methods compared to other baseline methods.

Quantization and online inference optimization. ColBERTv2 [23] and its highly optimized implementation PLAID [24] quantize every dimension of the encoded vector and represent the document as a summation from the clus-

ter centroid and the residuals. Residual compression can further reduce the index size while preserving retrieval quality. ColBERTv2 and PLAID also use an inverted index with clustering to improve online search speed. Our evaluation has built LeapMV on top of PLAID, taking advantage of its highly optimized implementation. DESSERT [6] proposes to speed up online query processing using a randomized algorithm for LSH-based vector set queries. DESSERT does not reduce storage space, and it uses much more space than ColBERTv2 or PLAID, though its online inference can be fairly fast. CITADEL [16] proposes to route each token vector to the predicted lexical "keys" such that a query token vector only interacts with document token vectors routed to the same key. While CITADEL is faster than PLAID and intends to use quantization, it has been reported to use more space than PLAID when desiring competitive relevance. EMVB [20] proposes an efficient query processing framework for multi-vector dense retrieval with centroid-based document pre-filtering and space compression with product quantization. XTR [15] approximates document rank scoring of multi-vector dense retrieval by retrieving the most important query-specific document tokens first and accounting for the contribution of the missing tokens to the overall score with missing similarity imputation.

LeapMV is orthogonal and complementary to the above online techniques because online optimization can be applied after offline processing with LeapMV's static token pruning of each document.

3 Learnable Pruning for Multi-Vector Representations

We now describe the LeapMV module, as illustrated in Figure 1. Let D be the sequence of M tokens representing each document d: $D = (D_1, D_2, \cdots, D_M)$, where $D_j \in \mathbb{R}^h$. To decide if the j-th token can be pruned or not, we let $D'_j \in \mathbb{R}^{(2c+1)\cdot h}$ represent the concatenation of a contextualized window with a window size of c before and after the j-th token. Namely, $D'_j = (D_{j-c} \circ \cdots \circ D_j \circ \cdots \circ D_{j+c})$, which forms a representation of total size $(2c+1)\cdot h$. As shown in Figure 1, the above representation is forwarded to one fully-connected neural network layer with dropout as a token-level classifier to obtain:

$$Y_j = W(D'_j \circ \xi) + b, \text{ with } j \in \{1, 2, \cdots, M\} \tag{2}$$

where W is a weight matrix transforming the inputs into classifier logits. $\xi \in \mathbb{R}^h$, is a vector of independent noise, drawn from a Bernoulli distribution with probability $1-p$ with p as dropout rate, and b is a bias term. Logits $Y_j \in \mathbb{R}^2$ denote classification scores for j-th token, i.e. dimension 0 is the unnormalized probability for keeping the token (decision [KEEP]) and dimension 1 the logit for dropping (decision [DROP]).

To obtain discrete decisions for pruning or keeping a token, we apply Straight-through Gumbel-Softmax [9, 10, 18] on Y_j, i.e.,

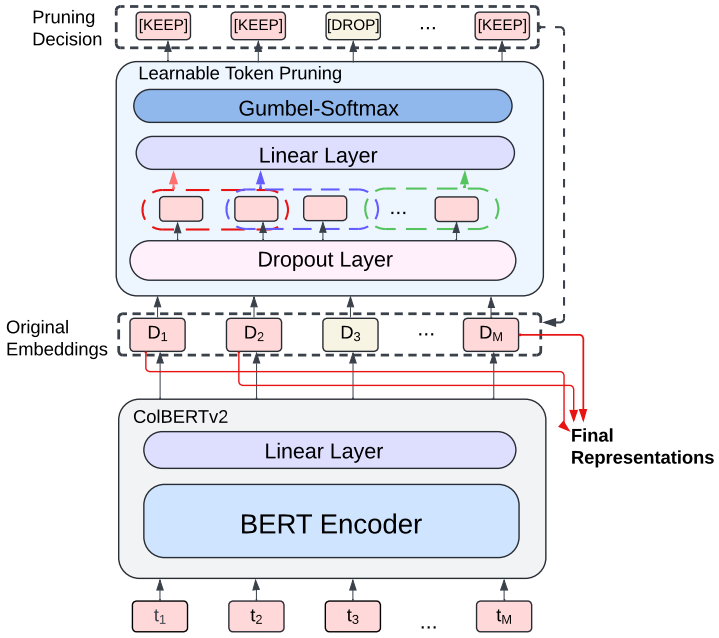

Fig. 1. LeapMV model architecture for pruning document tokens

$$(P_j)_i = \frac{\exp(\log((Y_j)_i + \epsilon_i)/\tau)}{\sum_{i'=0}^{1} \exp(\log((Y_j)_{i'} + \epsilon_{i'})/\tau)}. \tag{3}$$

$$u_j = \arg\max_{i \in \{0,1\}} (P_j)_i. \tag{4}$$

Here P_j is a vector with normalized probability, where $(P_j)_i$ is the i-th entry of vector P_j, and $(Y_j)_i$ is the i-th entry of vector Y_j. The hyperparameter τ is the softmax temperature, and ϵ_0, ϵ_1 are i.i.d samples from the Gumbel distribution $-\log(-\log(\text{Uniform}[0,1]))$. We obtain $u_j \in \{0,1\}$ as the prediction for the jth-token [KEEP] or [DROP] decision for document d.

Online inference. For scoring shown in Formula (1), only the token predicted as [KEEP] would be included for MaxSim computation. With token pruning under the predicted value u_j as 0 or 1 for j-token of a document, we compute the rank score of each document with a MaxSim operation as:

$$S_{q,d} = \sum_{i=1}^{N} \max_{j=1}^{M} Q_i \cdot ((1 - u_j) \cdot D_j^T). \tag{5}$$

In the above expression, tokens predicted as [DROP] are not used for calculating matching scores between queries and documents. The online inference complexity

is proportional to the average number of kept tokens instead of the original number of tokens.

Training. During model training, our scheme performs token-level classification, uses the classification results to perform modified late interaction, and receives feedback similar to the original training of ColBERTv2, which includes a knowledge distillation loss from the cross-encoder teacher as well as in-batch negative loss. During training with backpropagation, the Gumbel-SoftMax function provides a differentiable approximation to the argmax function that prunes tokens if they are not selected. It is known that the Gumbel-Softmax distribution is smooth for $\tau > 0$, yielding a well-defined gradient $\frac{\partial (P_j)_i}{\partial (I_j)_i}$ for the predicted softmax probability. Thus this is used to compute gradients, essentially by replacing discretely classified pruning choices with Gumbel-Softmax samples.

For low temperatures, the softmax computation of Formula (3) smoothly approaches the discrete argmax computation while preserving the relative order of the Gumbels $(\log(Y_j + \epsilon)/\tau)$. Samples from Gumbel Softmax distributions become near identical to samples from a [KEEP] and [DROP] class distribution. Thus the MaxSim scoring in training loss computation is approximated as

$$\tilde{S}_{q,d} = \sum_{i=1}^{N} \max_{j=1}^{M} Q_i \cdot ((P_j)_0 \cdot D_j^T). \tag{6}$$

Given a training dataset, let $\mathcal{V}^+$ be the subset of all positive documents for training query q, and $\mathcal{V}^-$ be a subset containing all negative documents for query q. The top one probability distribution of positive or negative document d_i is computed as:

$$\frac{\exp(\tilde{S}_{q,d_i})}{\sum_{d_j \in \mathcal{V}^+ \cup \mathcal{V}^-} \exp(\tilde{S}_{q,d_j})}.$$

To train the ColBERT model with the above token pruning classifier, we use the cross-entropy rank loss with KL divergence for knowledge distillation [23].

Notice that there is a difference from the original scoring Formula (1) to the actual online scoring Formula (5) due to token pruning, and there is a training approximation from Formula (1) to Formula (6) for pruning optimization through SGD learning. Our evaluation shows such an approximation leads to a limited relevance drop while gaining a large advantage for space and time-saving.

Adjustment with a space reduction target. Our evaluation finds that the above LeapMV module in Figure 1 yields approximately 50% token pruning. We can modify the Gumbel-Softmax probability outcome for each document towards a more or less aggressive token pruning target based on a desired space reduction goal. Let α be the target percentage of tokens to prune in this document.

Given the outcome of Formula (3) for each token in a document, we sort $(P_j)_0$ in a non-descending order where $1 \leq j \leq M$ for this document with M tokens. Without loss of generality, assume after such sorting, $(P_j)_0 \leq (P_{j+1})_0$ where $1 \leq j \leq M-1$. Pruning tokens from the sorted position 1 to x while

keeping all other tokens from position $x+1$ meets the target pruning ratio α. Then we set

$$\delta = 0.5 - 0.5((P_x)_0 + (P_{x+1})_0).$$

We can show that adding offset δ to the first dimension of the vectors (P_j) of this document will allow LeapMV to prune α percentage of tokens approximately. This is because the above offset δ satisfies

$$(P_j)_0 + \delta \leq (P_j)_1 - \delta \text{ where } 1 \leq j \leq x$$

and

$$(P_j)_0 + \delta \geq (P_j)_1 - \delta \text{ where } x+1 \leq j \leq M.$$

With the randomization introduced in Formula (3), the chance of being equal in the above inequalities is small. Then tokens from the sorted positions 1 to x are pruned approximately when applying Formula (4).

Thus to adjust the chance of being pruned towards target ratio α, we modify the vectors (P_j) for each document as follows after computation of Formula (3) and before computing Formula (4):

$$(P_j)_0 = (P_j)_0 + \delta \text{ and } (P_j)_1 = (P_j)_1 - \delta \text{ where } 1 \leq j \leq M.$$

The evaluation in Section 4 finds that while LeapMV cuts about 50% of tokens in its default setting without α adjustment, choosing α betwen 25% and 75% gives a range towards the desired space reduction target while incurring a small or modest relevance loss.

4 Evaluation

Datasets. To evaluate LeapMV for its out-of-domain retrieval quality, we perform experiments on two out-of-domain search data collections:

LoTTE [23], a collection of 12 test sets, focuses on information-seeking user queries and answers on diverse long-tail topics from StackExchange forums. Topics for each dataset include writing, recreation, science, technology, and lifestyle. Each topic has 400-2100 queries and 100k–2M passages. The pooled set with 2.8M passages contains the passages and queries aggregated across all topics to form a more diverse corpus. LoTTE also provides search and forum queries, gathered from filtered Google search queries [11] and StackExchange communities respectively. The performance is reported separately for each group.

BEIR collection [26] consists of 13 publicly available datasets on fact-checking, citation prediction, duplicate question retrieval, argument retrieval, news retrieval, question answering, tweet retrieval, bio-medical IR, and entity retrieval tasks. The size of these data sets varies from a few thousand to 5.4M.

Model training uses MS MARCO data [3] with 8.8M passages. We also report in-domain retrieval evaluations with the development set with 6980 queries.

Implementation. LeapMV is trained under the same setting as the ColBERTv2, such as the starting checkpoint, learning rates, the number of training

steps, and training data, with the exception of using a step accumulation of 2 to reach a batch size of 32. To be more specific, we initialize LeapMV with ColBERTv1.9 [4], the starting checkpoint released by ColBERT official repository, and train using four 40GB A100 GPUs for 400k steps. We train the model using 64-way mined hard negatives provided by ColBERT, consisting of a query, a highly-ranked passage (or gold labels), and one or more negative passages with the same cross-encoder teacher [19]. For indexing, we follow the default PLAID setting of hyper-parameters when the search depth is 1000 unless otherwise mentioned. We adopt a dropout rate of $p = 0.1$ for Formula (2) and temperature $\tau = 2/3$ for Formula (3), similar as [5, 18].

Metrics. We follow the standard practice in the literature, reporting, success@5 (recall@5) for LoTTE, NDCG@10 for BEIR, and mean reciprocal rank at 10 (MRR@10) and recall at 1000 for MS MARCO Dev set.

For efficiency, we report the total amount of in-memory space in gigabytes needed to host the search data and the mean latency to execute a test query in milliseconds with a retrieval depth of 1000. For GPU latency, we measure on one NVIDIA A40 GPU and allow full access to 16 threads available. For CPU latency, we measure on an AMD EPYC 7763 (AMD Milan) running with 8 threads or 1 thread. On average, the 8-thread CPU time is about 3.7x faster than the 1-thread CPU. We report the 8-thread CPU time.

Baselines. We compare several baselines discussed in Section 2, which directly reduces the number of tokens in the ColBERT multi-vector representation.

- **Text chunking**. We will compare two text chunking methods: phrase-based and sentence-based chunking. Both methods aggregate the embeddings for each phrase or each sentence by mean pooling. We use NLTK-tagger-chunker as labels of boundaries for each phrase and use NLTK sentence tokenization package to obtain sentence partitions [2].
- **Static pruning**. We have implemented the first-k and attention-k methods of top k pruning [14], where we keep the first k tokens of a document or the top k tokens that receive the most amount of attention in the last layer of the document encoder. We also implemented IDF-uniform with parameter IDF threshold τ [1], which is the recommended technique from the paper. IDF parameter τ means to prune the top-τ tokens with the highest IDF value in MS MARCO corpus uniformly in all documents.

4.1 Zero-shot retrieval performance in out-of-domain datasets

We evaluate LeapMV outside the training domain to analyze its zero-shot effectiveness against pruning baselines and some state-of-the-art dense retrieval models.

Table 1 presents, under the same relevance accuracy constraints, the index space and latency requirements for different pruning methods in LoTTE pooled dataset. Accuracy 98% indicates that the method retains at least 98% of Success@5 performance compared to PLAID ColBERTv2. Each table entry lists the minimum space size (in GB) and GPU latency (in ms) for each method after pruning, based the PLAID framework, to meet or exceed the specified accuracy.

Table 1. Space and latency for LoTTE pooled dataset under different accuracy constraints. PLAID ColBERTv2 takes 19.6 ms for the search queries and 24.8 ms for the forum queries with an index size of 13 GB.

Accuracy	98%		97%		95%		93%	
(%)	Space	Latency	Space	Latency	Space	Latency	Space	Latency
	Search test queries							
First	-	-	12.0 (2.3×)	11.9 (1.5×)	9.2 (2.6×)	10.3 (1.5×)	6.9 (2.5×)	9.0 (1.5×)
Attn	10.0 (1.5×)	10.6 (1.2×)	8.4 (1.6×)	9.8 (1.2×)	6.8 (1.9×)	8.8 (1.3×)	5.9 (2.1×)	8.0 (1.3×)
IDF	-	-	8.8 (1.7×)	20.7 (2.6×)	7.5 (2.1×)	17.8 (2.7×)	6.2 (2.2×)	16.8 (2.7×)
LeapMV	**6.5**	**8.5**	**5.2**	**8.0**	**3.5**	**6.7**	**2.8**	**6.2**
	Forum test queries							
First	-	-	-	-	11.0 (3.1×)	13.3 (1.8×)	9.2 (3.3×)	12.2 (1.8×)
Attn	11.0 (1.7×)	13.8 (1.4×)	10.0 (1.7×)	13.0 (1.4×)	7.6 (2.2×)	10.7 (1.4×)	5.9 (2.1×)	8.8 (1.3×)
IDF	8.8 (1.4×)	24.0 (2.4×)	8.0 (1.4×)	21.7 (2.3×)	7.2 (2.1×)	20.8 (2.8×)	6.2 (2.2×)	18.3 (2.7×)
LeapMV	**6.5**	**10.1**	**5.9**	**9.4**	**3.5**	**7.4**	**2.8**	**6.7**

Table 2. Relevance for LoTTE pooled dataset under space budgets (percentage of the PLAID space size). Entries are best Success@5 one method achieves under a space constraint.

Space budget	75%	50%	40%	32.5%	25%	17.5%
LoTTE search queries	Maximum Success@5					
First	68.3	65.3	63.1	61.1	57.8	53.8
Attn	70.6	67.9	66.6	65.1	60.5	49.8
IDF-uniform	70.3	67.5	62.6	60.7	59.5	-
LeapMV	**70.7**	**70.6**	**70.0**	**69.4**	**67.8**	**66.1**
LoTTE forum queries	Maximum Success@5					
First	58.9	56.3	55.2	53.1	50.3	45.6
Attn	61.8	59.1	58.1	56.0	51.8	43.0
IDF-uniform	62.5	59.9	56.3	54.0	51.8	48.8
LeapMV	**62.7**	**62.2**	**61.4**	**60.9**	**59.8**	**57.5**

To meet the desired accuracy, we adjust k for the first-k and attention-k methods, IDF threshold τ in IDF-uniform, and α pruning ratio in LeapMV. The entry '-' means that such a method did not deliver the corresponding accuracy under the tested settings. Table 1 demonstrates that, under the same relevance constraint, LeapMV requires significantly less space and operates much faster. For example, with less than 2% loss (98% accuracy), LeapMV achieves 2× reduction in space and 2.5× speedup on GPU compared to PLAID. To achieve 97% of accuracy, LeapMV requires 5.2GB for search test queries, while IDF-uniform uses 1.7× more space and takes 2.6× longer.

Table 2 lists the retrieval quality, measured with Success@5 in LoTTE, under the same space budget constraint. Space size 75% means to have the index of at most about 75% of original ColBERTv2 PLAID, which is approximately 13GB for LoTTE pooled dataset. For example, using about 50% of the space ColBERTv2 PLAID needs (about 6.5GB), LeapMV can deliver Success@5 of 70.6 while other methods can deliver at most 67.9. When adhering to a tight space budget, such as 17.5% of the original index, the baseline methods experience a

Table 3. Zero-shot retrieval performance on LoTTE test set benchmark.

	Rocket -QAv2	Retro -MAE	PLAID	First k=50	Attn k=50	IDF τ=100	IDF τ=300	LeapMV
Space	-	-	13GB	5.1GB	5.1GB	8GB	6.2GB	6.5GB
	LoTTE search test queries (Success@5)							
Writing	78.0	-	80.6	74.2	76.3	76.5	74.3	76.2
Recreation	72.1	-	72.3	64.1	67.2	72.4	71.4	72.2
Science	55.3	-	56.7	49.8	53.2	55.4	54.3	55.3
Technology	63.4	-	66.1	56.5	59.4	63.6	62.6	67.1
Lifestyle	82.1	-	84.3	73.1	80.9	83.4	82.5	84.7
Pooled	69.8	66.8	72.1	63.1	66.6	69.3	67.5	70.2
% Loss	-	-	0%	12.5%	7.6%	3.9%	6.4%	2.6%
	LoTTE forum test queries (Success@5)							
Writing	71.5	-	76.2	70.9	73.8	74.0	72.0	75.8
Recreation	65.7	-	71.4	60.4	66.5	70.9	69.3	69.6
Science	38.0	-	47.2	39.0	39.9	45.9	44.1	44.4
Technology	47.3	-	53.7	48.3	46.8	52.1	49.9	53.3
Lifestyle	73.7	-	76.9	68.0	72.7	76.5	75.2	77.1
Pooled	57.7	58.5	63.5	55.2	58.1	62.1	59.9	62.1
% Loss	-	-	0%	13.1%	8.5%	2.2%	5.7%	2.2%

decrease in performance from 25-31% compared to PLAID in search queries. In contrast, LeapMV only shows an 8.3% decline. Overall, LeapMV consistently delivers better relevance under the same space budget and is robust under more aggressive compression ratio.

To assess the performance for each subtopics of LoTTE, Table 3 presents the Success@5 (Recall@5) of different methods with several parameter settings with index sizes comparable to our default LeapMV. As a point of reference, we also include the average NDCG@10 of two recent single-vector dense retrievers RetroMAE [17] and RocketQAv2 [21]. The result from Table 3 shows that LeapMV can deliver about 2× reduction in space usage while incurring only 2.6% and 2.2% relevance loss. In comparison, IDF-uniform with τ=100 can deliver visibly smaller 1.6× space reduction while having a larger average loss of relevance (3.9% and 2.2%). These findings are consistent with the results discussed in Table 1 and Table 2.

Table 4 shows the zero-shot retrieval performance, measured in NDCG@10, across 13 BEIR datasets for out-of-domain search. For relevance comparison, we also list NDCG@10 of single-vector dense retrievers RetroMAE and RocketQAv2. The targeted space reduction is about 50% of the PLAID ColBERTv2 size, achieved by selecting $k = 50$ for top-k pruning and τ=1000 for IDF. By default, LeapMV reduces space usage by about 2×, with only a 3.28% relevance loss compared to ColBERTv2 PLAID. In comparison, the first-k, attention-k, and IDF uniform have an NDCG@10 loss varying from 8.98% to 16%.

Table 4. Zero-shot retrieval performance on 13 BEIR datasets.

BEIR Datasets	Rocket-QAv2	Retro-MAE	PLAID	First-k	Attn-k	IDF-τ	LeapMV
% of PLAID space	-	-	100%	54%	54%	49%	49%
SciFact	56.8	65.3	69.3	54.9	47.5	62.9	68.9
NFCorpus	29.3	30.8	33.8	30.7	26.5	31.9	34.1
ArguAna	45.1	43.3	46.3	44.5	45.1	41.4	44.9
SCIDOCS	13.1	15.0	15.4	14.8	13.9	14.6	15.2
Touche-2020	24.7	23.7	26.3	22.8	22.1	26.2	24.0
FiQA	30.2	31.6	35.6	26.2	30.0	30.3	34.6
T-COVID	67.5	77.2	73.8	68.5	55.2	61.7	75.2
NQ	50.5	51.8	56.1	48.5	48.4	48.8	54.5
DBPedia	35.6	39.0	44.6	42.5	42.6	41.6	42.5
HotpotQA	53.3	63.5	66.7	57.8	58.5	62.6	59.6
FEVER	67.6	77.4	78.5	63.3	56.5	74.8	74.1
C-FEVER	18.0	23.2	17.6	13.7	11.5	17.6	18.1
Quora	74.9	84.7	85.2	85.4	85.4	76.9	82.5
Average NDCG@10	43.6	48.2	50.0	44.1	41.8	45.5	48.3
% Loss	-	-	0%	11.69%	16.35%	8.98%	3.28%

4.2 In-domain search with MS MARCO

Table 5 lists the performance of LeapMV for MS MARCO passage ranking in comparison with the original PLAID ColBERTv2, phrase chunking, sentence chunking, top-k static pruning, and IDF-uniform. The results from Table 5 show the relevance of LeapMV is fairly close to static top-k pruning which uses a fixed top k parameter. LeapMV incurs a relevance loss of 2.7% for MRR@10 and 1.6% for Recall@1K compared to the original PLAID implementation of ColBERTv2, while achieving approximately 2× space reduction and 1.8× faster GPU latency. In contrast, IDF uniform with $\tau = 100$ uses 33% more space than LeapMV default while their Recall@1K difference is within 1%. When increasing τ to 3000, IDF's space cost 5.4GB approaches LeapMV with $\alpha = 80\%$, but results in a rapid decline in relevance, making it substantially less effective than LeapMV. Similar conclusions can be drawn with the first k and the attention k method, demonstrating the robustness of LeapMV. The two text chunking methods are not competitive due to their significant losses in retrieval performance.

As a reference, we also list the relevance performance of single-vector dense retrieval RetroMAE and RocketQAv2 [21]. The released RetroMAE checkpoint gives 0.359 MRR@10 using the standard MS MARCO, and this is below 0.416 reported in [17]. The reason was explained in [13] that, following RocketQAv2 [21], the paper evaluates the modified MS MARCO dataset with title annotation, which is not fair. Since the original dataset released does not utilize title information, all experiments in ColBERT and our evaluation follows the standard approach to use the original MS MARCO without title annotation.

Table 5. End-to-end in-domain retrieval performance for MS MARCO passages.

Methods	Relevance		Latency ms (Speedup)				Space	
	MRR@10	R@1K	GPU		8-thr. CPU		GB	(Ratio)
RocketQA	38.8	98.1	-	-	-	-	27	-
RetroMAE (no title anno.)	36.0	97.7	-	-	-	-	27	-
PLAID ColBERTv2	39.4	97.6	18.9	(1×)	69.1	(1×)	22	(1×)
Phase Chunking	30.0	89.6	11.5	(1.6×)	56.5	(1.2×)	12	(1.8×)
Sentence Chunking	22.0	69.4	5.6	(3.4×)	16.3	(4.2×)	2.1	(10.5×)
First k=15	31.7	84.7	5.9	(3.2×)	30.6	(2.3×)	4.9	(4.5×)
First k=50	38.0	95.8	8.7	(2.2×)	59.8	(1.2×)	16	(1.4×)
Attn k=15	28.2	86.4	5.7	(3.3×)	29.1	(2.4×)	4.8	(4.6×)
Attn k=50	38.6	97.1	8.6	(2.2×)	62.2	(1.1×)	16	(1.4×)
IDF τ=3000	27.3	86.1	8.9	(2.1×)	27.3	(2.5×)	5.4	(4.1×)
IDF τ=100	38.5	97.0	13.0	(1.5×)	54.6	(1.3×)	15	(1.5×)
LeapMV, α=0.8	35.8	93.6	6.0	(3.2×)	26.1	(2.6×)	4.9	(4.5×)
LeapMV, default	38.4	96.0	8.0	(2.4×)	49.9	(1.4×)	12	(1.8×)
LeapMV, α=0.25	38.8	96.2	9.5	(2.0×)	59.5	(1.2×)	16	(1.4×)

4.3 Ablation studies

We present two ablation studies regarding model architecture design.

Impact of window size choices. Table 6 shows the impact of the window size c in Formula (2) on the MS MARCO passage, Dev set. Due to the cost of model training, we train for 300k steps instead of 400k steps for this study. The result shows that imposing a wider context window does not yield much performance gain. We suspect that document token encodings already contain sufficient contextualized information in MS MARCO. For simplicity, all evaluations including zero-shot retrieval use $c = 0$.

Table 6. Impact of window size choice c on MS MARCO dev set.

c = 0		c = 2		c = 5	
MRR@10	Recall@1K	MRR@10	Recall@1K	MRR@10	Recall@1K
38.3	96.0	38.2	96.2	38.2	95.8

Linear layer versus transformer in LeapMV. Figure 1 uses a linear layer to compute the pruning decision. Table 7 shows the impact of replacing a single linear layer with a more complex transformer encoder [27]. For the transformer encoder, hidden dimension d is equal to the default ColBERT encoding of 128, 8 attention heads, and feed-forward layer dimension of $4d$. Despite integrating this more complex module, both modules yield a similar relevance. For better indexing efficiency, we choose the linear layer with fewer parameters for LeapMV.

Table 7. Choice of linear layer versus transformer network on MS MARCO dev set for computing pruning decisions in LeapMV architecture.

Linear Layer		Transformer	
MRR@10	Recall@1K	MRR@10	Recall@1K
38.4	96.0	38.4	95.9

5 Concluding Remarks

We introduce LeapMV, a learned token pruning scheme designed for efficient multi-vector dense retrieval with a moderate relevance tradeoff to enhance efficiency. LeapMV makes a context-aware pruning decision for each document and is more adaptive than existing baselines. The degree of pruning in LeapMV can be adjusted with parameter α to meet the targeted space reduction goals, resulting in a near-linear reduction in both GPU and CPU latency.

Our evaluation shows the advantage of LeapMV over the previous baselines in adapting to various space reduction goals. For instance, Table 1 shows that under the same relevance accuracy budget, LeapMV consumes up to 3.3× less space and is up to 2.8× faster for LoTTE than top-k token pruning and IDF methods. Compared to the optimized PLAID ColBERTv2, LeapMV achieves 2× space saving and 2.5× GPU latency speedup while incurring less than 2% relevance loss on LoTTE forum queries. LeapMV yields more space and time saving when accuracy can be relaxed further while maintaining performance significantly better than its competitors under tight space budgets. Our current evaluation follows the default parameter setting in PLAID with a retrieval depth of 1000, and a future study is to investigate the impact of varying these settings.

Acknowledgments. We thank anonymous referees for their valuable comments. The major portion of this work was supported by Amazon. It was partially supported by U.S. NSF IIS-2225942 and its ACCESS program in using its computing resource. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Amazon or the NSF.

References

1. Acquavia, A., Macdonald, C., Tonellotto, N.: Static pruning for multi-representation dense retrieval. In: Proceedings of the ACM Symposium on Document Engineering 2023. DocEng '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3573128.3604896, https://doi.org/10.1145/3573128.3604896
2. Bird, S., Loper, E.: NLTK: The natural language toolkit. In: Proceedings of the ACL Interactive Poster and Demonstration Sessions. pp. 214–217. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), https://aclanthology.org/P04-3031
3. Campos, D.F., Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L., Mitra, B.: Ms marco: A human generated machine reading comprehension dataset. ArXiv **abs/1611.09268** (2016)
4. colbertv1.9: https://huggingface.co/colbert-ir/colbertv1.9 (2022)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
6. Engels, J., Coleman, B., Lakshman, V., Shrivastava, A.: DESSERT: An efficient algorithm for vector set search with vector set queries. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=kXfrlWXLwH
7. Formal, T., Lassance, C., Piwowarski, B., Clinchant, S.: From distillation to hard negative sampling: Making sparse neural IR models more effective. SIGIR (2022)
8. Formal, T., Piwowarski, B., Clinchant, S.: SPLADE: Sparse lexical and expansion model for first stage ranking. SIGIR (2021)
9. Gumbel, E.: Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures. Applied mathematics series, U.S. Government Printing Office (1954)
10. Jang, E., Gu, S.S., Poole, B.: Categorical reparameterization with gumbel-softmax. ICLR (2017)
11. Khashabi, D., Ng, A., Khot, T., Sabharwal, A., Hajishirzi, H., Callison-Burch, C.: Gooaq: Open question answering with diverse answer types (2021), https://arxiv.org/abs/2104.08727
12. Khattab, O., Zaharia, M.A.: Colbert: Efficient and effective passage search via contextualized late interaction over bert. SIGIR (2020)
13. Lassance, C., Clinchant, S.: The tale of two msmarco - and their unfair comparisons. In: Proceed. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 2431–2435. SIGIR '23, ACM, New York, NY, USA (2023)
14. Lassance, C., Maachou, M., Park, J., Clinchant, S.: Learned token pruning in contextualized late interaction over bert (colbert). In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 2232–2236. SIGIR '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3477495.3531835, https://doi.org/10.1145/3477495.3531835
15. Lee, J., Dai, Z., Duddu, S.M.K., Lei, T., Naim, I., Chang, M.W., Zhao, V.Y.: Rethinking the role of token retrieval in multi-vector retrieval. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=ZQzm0Z47jz

16. Li, M., Lin, S.C., Oguz, B., Ghoshal, A., Lin, J., Mehdad, Y., tau Yih, W., Chen, X.: CITADEL: Conditional token interaction via dynamic lexical routing for efficient and effective multi-vector retrieval. pp. 11891–11907 (2023)
17. Liu, Z., Xiao, S., Shao, Y., Cao, Z.: RetroMAE-2: Duplex masked auto-encoder for pre-training retrieval-oriented language models. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2635–2648. Association for Computational Linguistics, Toronto, Canada (Jul 2023)
18. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: A continuous relaxation of discrete random variables. ICLR (2017)
19. MiniLM-L-6-v2: https://huggingface.co/cross-encoder/ms-marco-minilm-l-6-v2 (2022)
20. Nardini, F.M., Rulli, C., Venturini, R.: Efficient multi-vector dense retrieval with bit vectors. In: Goharian, N., Tonellotto, N., He, Y., Lipani, A., McDonald, G., Macdonald, C., Ounis, I. (eds.) Advances in Information Retrieval (ECIR 2024). pp. 3–17. Springer Nature Switzerland, Cham (2024)
21. Ren, R., Qu, Y., Liu, J., Zhao, W.X., She, Q., Wu, H., Wang, H., Wen, J.R.: RocketQAv2: A joint training method for dense passage retrieval and passage reranking. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 2825–2835. ACM, Online and Punta Cana, Dominican Republic (Nov 2021)
22. Sang, E.F.T.K., Buchholz, S.: Introduction to the conll-2000 shared task: Chunking (2000), https://arxiv.org/abs/cs/0009008
23. Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.A.: Colbertv2: Effective and efficient retrieval via lightweight late interaction. ArXiv **abs/2112.01488** (Dec 2021)
24. Santhanam, K., Khattab, O., Potts, C., Zaharia, M.: Plaid: An efficient engine for late interaction retrieval. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. p. 1747–1756. CIKM '22 (2022)
25. Sciavolino, C., Zhong, Z., Lee, J., Chen, D.: Simple entity-centric questions challenge dense retrievers. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 6138–6148. ACL (Nov 2021)
26. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In: NeurIPS (2021)
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023), https://arxiv.org/abs/1706.03762
28. Yang, Y., Qiao, Y., He, S., Yang, T.: Weighted kl-divergence for document ranking model refinement. arxiv and SIGIR 2024 (2024)