Investigating Multi-layer Representations for Dense Passage Retrieval

Zhongbin Xie[1], **Thomas Lukasiewicz**[2,1]
[1] University of Oxford, UK [2] Vienna University of Technology, Austria
zhongbin.xie@cs.ox.ac.uk, thomas.lukasiewicz@tuwien.ac.at

Abstract

Dense retrieval models usually adopt vectors from the last hidden layer of the document encoder to represent a document, which is in contrast to the fact that representations in different layers of a pre-trained language model usually contain different kinds of linguistic knowledge, and behave differently during fine-tuning. Therefore, we propose to investigate utilizing representations from multiple encoder layers to make up the representation of a document, which we denote Multi-layer Representations (MLR). We first investigate how representations in different layers affect MLR's performance under the multi-vector retrieval setting, and then propose to leverage pooling strategies to reduce multi-vector models to single-vector ones to improve retrieval efficiency. Experiments demonstrate the effectiveness of MLR over dual encoder, ME-BERT and ColBERT in the single-vector retrieval setting, as well as demonstrate that it works well with other advanced training techniques such as retrieval-oriented pre-training and hard negative mining.

1 Introduction

Dense passage retrieval is adopted in open-domain question answering (Lee et al., 2019; Karpukhin et al., 2020) to retrieve relevant passages from a large corpus for the reader model to extract answers. The dense retrieval technique encodes queries and documents into dense embeddings, and has wide applications in other knowledge-intensive tasks (Petroni et al., 2021) as well as retrieval-augmented generation (RAG, Lewis et al., 2020; Zhao et al., 2024). Dense retrieval enjoys many advantages over sparse retrieval methods (Sparck Jones, 1972; Robertson and Zaragoza, 2009), such as alleviation of the term mismatch problem (Furnas et al., 1987), and improved retrieval performance through supervised learning (Karpukhin et al., 2020).

Dense retrieval models typically adopt a dual-encoder architecture (Karpukhin et al., 2020, cf. Figure 1 (a)), where the query and document are encoded by two encoders usually fine-tuned from a pre-trained language model. However, current dense retrieval architectures (Khattab and Zaharia, 2020; Zhang et al., 2022b; Wu et al., 2022) represent documents with vectors[1] taken only from the encoder's last hidden layer (Figure 1 (b)). This is in contrast to studies which have shown that, for a pre-trained language model like BERT (Devlin et al., 2019), representations in different layers contain different kinds of linguistic knowledge, and behave differently during fine-tuning (Rogers et al., 2020). For example, syntactic information resides mainly in the middle layers of BERT, while the semantics spreads across all the layers (Hewitt and Manning, 2019; Jawahar et al., 2019; Tenney et al., 2019), and the final layers are most task-specific after fine-tuning (Liu et al., 2019; Kovaleva et al., 2019; Hao et al., 2019). Given this observation, we propose to investigate utilizing representations from multiple encoder layers, instead of those only from the last layer, to make up the representation of a document, which we denote Multi-layer Representations (MLR, Figure 1 (c)).

A straightforward way to utilize representations from multiple encoder layers to represent a document is to retrieve in a multi-vector setting, which is illustrated in Figure 1 (c). Unlike the vanilla dual-encoders where each document is represented by a *single* vector, multi-vector models (Khattab and Zaharia, 2020; Luan et al., 2021; Zhang et al., 2022b; Wu et al., 2022) represent each document with *multiple* vectors, and thus enjoy more representational capacity. We propose to investigate how representations in different layers affect this representational capacity, and how this compares

[1] We use the term "embedding" and "vector" interchangeably in this paper.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 24522–24536
November 4-9, 2025

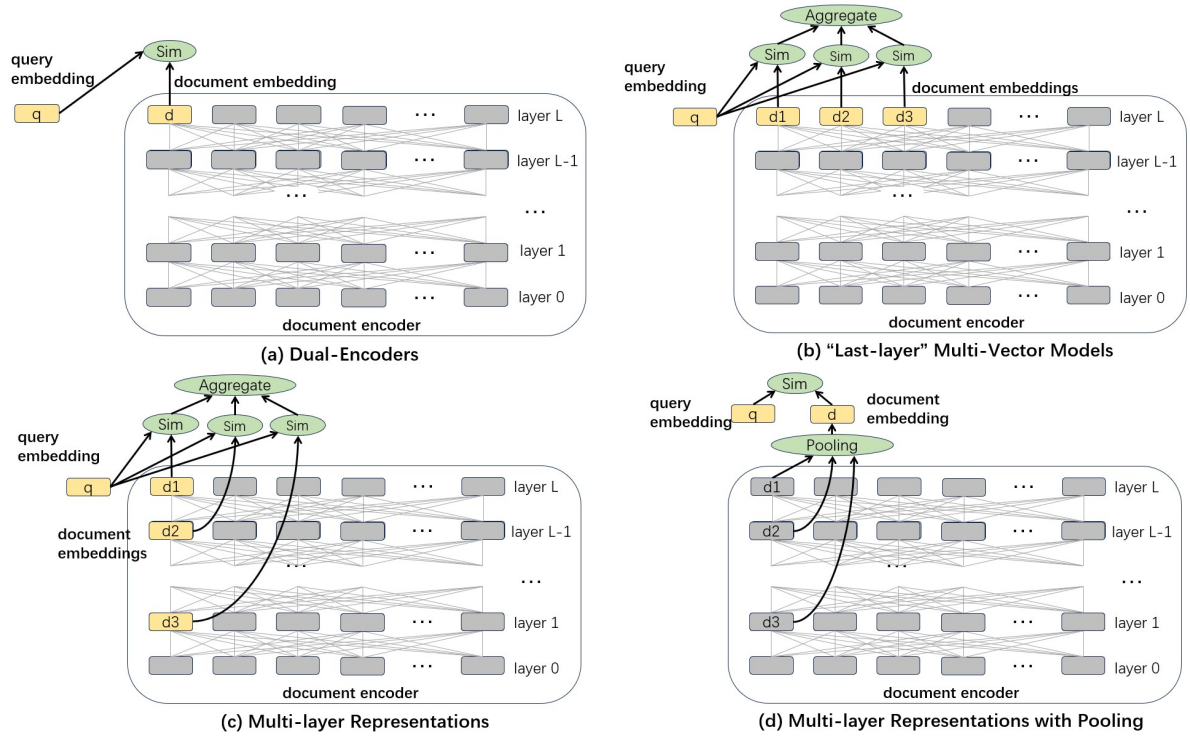

Figure 1: Illustrations of different dense retrieval models. Yellow boxes represent query/document embedding vectors; green ellipses represent operations. Details of (c) and (d) are described in § 3.

to previous "last-layer" multi-vector models like ME-BERT (Luan et al., 2021).

On the other hand, multi-vector models' improved retrieval performance over single-vector models usually comes with the cost of a decreased retrieval efficiency, because the number of document embeddings to be searched is proportional to the average number of vectors used to represent each document. For example, for an 8-vector ME-BERT (Luan et al., 2021), it takes 481GB disk space to store the document embeddings for ∼21M documents, as well as 3.0321 seconds per query to build and retrieve the entire index. In contrast, for a single-vector dual-encoder, it only takes 60GB and 0.2056 seconds on the same machine. Given this, we further propose to pool the multi-vector representation of a document into a single one so that multi-vector models can be reduced to single-vector models during inference time (Figure 1 (d)). Thus, we can enjoy an improved retrieval performance over a single-vector dual-encoder with exactly the same retrieval efficiency.

Our contributions are as follows:

- We propose to utilize representations from multiple encoder layers to represent a document, which is denoted as Multi-layer Representations (MLR). We investigate how representations in different layers affect MLR's performance under the multi-vector retrieval setting, and find that, with the last few layers and a relatively small vector number, MLR can effectively outperforms baselines with both BERT and T5. But unlike ME-BERT, MLR's representational capacity cannot scale up well with more representation vectors.
- We further propose to leverage pooling strategies to reduce multi-vector models to single-vector ones to improve retrieval efficiency. We explore self-contrastive pooling, average pooling, and scalar mix pooling, and find that single-vector MLR can outperform a single-vector dual encoder by a large margin.
- We demonstrate on diverse in-domain and out-of-domain retrieval datasets that single-vector MLR works well with other advanced training techniques such as retrieval-oriented pre-training and hard negative mining.[2]

2 Related Work

Single-vector retrieval models. DPR (Karpukhin et al., 2020) adopts a dual-encoder architecture

[2]The code of this paper is available at https://github.com/x-zb/mlr.

and a contrastive loss to learn dense representations for queries and documents. Improved training techniques are further developed to learn better single-vector representations, and can be roughly divided into three categories: (i) hard negative mining (Xiong et al., 2021; Sun et al., 2022), (ii) retrieval-oriented pre-training (Gao and Callan, 2021, 2022; Xiao et al., 2022; Liu et al., 2023), and (iii) knowledge distillation from cross encoders (Hofstätter et al., 2020; Qu et al., 2021; Ren et al., 2021; Tao et al., 2024). These techniques are in general orthogonal to our proposed method, and we empirically investigate MLR's compatibility with retrieval-oriented pre-training and hard negative mining in § 4.3.2.

Multi-vector retrieval models. Dual-encoders represent each document as a single vector, thus have limited representational power for long documents (Luan et al., 2021), are prone to be overfitting (Menon et al., 2022), and struggle to handle the one-to-many scenario where one document contains answers to multiple different queries (Zhang et al., 2022b; Wu et al., 2022). Therefore, several multi-vector models have been proposed. Specifically, ColBERT (Khattab and Zaharia, 2020; Santhanam et al., 2022) adopts representations of all the tokens in a document, while ME-BERT (Luan et al., 2021) and MVR (Zhang et al., 2022b) adopt a fixed number of vectors which are much fewer than the document length. Tang et al. (2021) cluster the token representations and adopt the resulting cluster centers to represent the document. Wu et al. (2022) segment a document into sentences and for each sentence introduce a learnable token, whose representations are then used to represent the document. All these models use representations from the document encoder's last layer, while we propose to explore the performance of intermediate layers.

Utilizing intermediate layers in deep learning. There are also related work on utilizing multiple hidden layers of a neural network to make predictions (Yan et al., 2015; Huang et al., 2018; Wehrmann et al., 2018; Manginas et al., 2020; Evci et al., 2022). For example, Manginas et al. (2020) leverage different layers of BERT representations for hierarchical multi-label document classification, while Hosseini et al. (2023) investigate BERT layers combination for semantic textual similarity. In information retrieval, Nie et al. (2018) aggregate the matching score of a query-document pair from different layers of a convolutional neural network, but their model focuses on the reranking task, and cannot be applied to large-scale retrieval. Ennen et al. (2023) leverage hierarchical representations of a BERT encoder to represent a query (not a document as we do), and require to dynamically adjust the document index during search. Moreover, they report negative results on dense passage retrieval. In contrast, we directly leverage multi-layer representations from a pre-trained language model to represent a document, and demonstrate its effectiveness in different scenarios.

3 Multi-layer Representations

Given a text query q, we aim to find the relevant documents from a large document collection $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, where N can range from millions to billions. We adopt one query encoder $E_Q(\cdot)$ and one document encoder $E_D(\cdot)$ to represent the corresponding text sequences as real-valued dense vectors.. For the query/document encoders, we will experiment with two pre-trained language models, BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020). For convenience, we will describe our method using BERT, and highlight the adaptations for T5 in § 3.3.

For a query q, the *[CLS]* representation in the last layer is adopted as the query embedding $E_Q(q) \doteq h_q \in \mathbb{R}^D$. For a document d with T tokens, assume the output of the document encoder with L transformer layers is a set of hidden states $\{h_i^{(l)} \in \mathbb{R}^D \mid l = 0, 1, 2, \ldots, L; i = 0, 1, 2, \ldots, T\}$, where $h_i^{(l)}$ denotes the hidden state in layer l at position i (layer 0 denotes the word embedding layer, and position 0 denotes the *[CLS]* token). For a dual-encoder, $E_D(d) \doteq h_0^{(L)}$; for ME-BERT (Luan et al., 2021), $E_D(d) \doteq \{h_i^{(L)} \mid i = 0, 1, \ldots, m-1\}$, where m is the number of representation vectors for each document; for ColBERT (Khattab and Zaharia, 2020), $E_D(d) \doteq \{h_i^{(L)} \mid i = 0, 1, \ldots, T\}$.

3.1 Multi-vector retrieval

BERT's representations in different layers contain different kinds of knowledge and behave differently during fine-tuning. Thus, to enrich our document representation, we leverage the *[CLS]* representations $h_0^{(l)}$ from different layers to represent a document, as shown in Figure 1 (c). Specifically, we adopt

$$E_D(d) \doteq \{h_0^{(l)} \mid l \in \mathcal{S} = \{l_1, l_2, \ldots, l_m\}\} \quad (1)$$

as our multi-vector representation for a document d. Here, $\mathcal{S} = \{l_1, l_2, \ldots, l_m\} \subseteq \{1, 2, \ldots, L\}$, and we always include the last layer (i.e., $l_m = L$) to leverage the full depth of the encoder.

We define the similarity between a query and a document as the maximum inner product between the query embedding h_q and all of the document embeddings in $E_D(d)$:

$$\text{sim}(q, d) = \max_{h_0^{(l)} \in E_D(d)} h_q^\intercal h_0^{(l)}. \quad (2)$$

The number of representation vectors per document (i.e., m), as well as how to choose the specific layers for a fixed m, are the keys for model performance. In § 4.2, we will investigate different strategies including using last few layers, first few layers, and uniformly distributed layers.

Training. Given a training instance $\langle q, d^+, d_1^-, \ldots, d_n^- \rangle$, where d^+ is a positive (relevant) document, and the d_i^-s are n negative (irrelevant) documents, we adopt the following contrastive loss to optimize encoder parameters:

$$\mathcal{L}(q, d^+, d_1^-, \ldots, d_n^-) = -\log \frac{e^{\text{sim}(q,d^+)}}{e^{\text{sim}(q,d^+)} + \sum_{i=1}^n e^{\text{sim}(q,d_i^-)}}. \quad (3)$$

In practice, we follow Karpukhin et al. (2020) to include one negative passage for each query, and adopt the in-batch negatives technique, where all the (positive and negative) documents corresponding to other queries in the same mini-batch are used as negative documents for this query. Thus, the number of negatives n in Eq. (3) equals to $2(B-1)+1$ with B being the batch size.

Inference. We adopt the `FAISS` library (Johnson et al., 2021) to index all the document vectors and conduct nearest neighbor search. Unlike models like ColBERT (Khattab and Zaharia, 2020) and DCSR (Wu et al., 2022) where the number of representation vectors is different for each document, our model adopts a fixed number of vectors m. So, when retrieving top-k documents, we can simply retrieve top-km vectors, and map them back to document ids by dividing each vector id with m and taking the integer part. Finally, since we adopt the maximum dot product in Eq. (2), we can just merge the same document ids and take the top-k results.

3.2 Reducing to single-vector models through pooling

We further investigate if pooling multiple vectors to a single vector can achieve an improved performance in the single-vector retrieval setting. This is desirable, since single-vector retrieval is much more efficient than multi-vector retrieval, in terms of both space and time complexity, because the number of document embeddings to be searched is mN, which is proportional to the number of vectors used to represent each document.

Recall that $E_D(d) = \{h_0^{(l)} \mid l \in \mathcal{S}\}$ is a set of m vectors that we use to represent a document, and assume that $h_D(d)$ is the single vector pooled from $E_D(d)$ to represent a document during inference. We propose the following **self-contrastive pooling** strategy:

During inference, we simply take the last layer *[CLS]* representation $h_0^{(L)}$ as the single vector representation for a document:

$$h_D(d) = h_0^{(L)}. \quad (4)$$

During training, we adopt the same maximum inner product similarity as Eq. (2) for negative documents, but use the inner product of h_q and $h_D(d^+)$ for positive documents, resulting in the following contrastive loss:

$$\mathcal{L}_{con}(q, d^+, d_1^-, \ldots, d_n^-) = -\log \frac{e^{h_q^\intercal h_D(d^+)}}{e^{h_q^\intercal h_D(d^+)} + \sum_{i=1}^n e^{\text{sim}(q,d_i^-)}}. \quad (5)$$

Here, for positive documents, the vector whose inner product with h_q is maximized is the same with that used during inference (i.e., $d_D(d^+)$). For negative documents, since we are minimizing the maximum inner product, the inner product for $h_D(d^-)$ is also minimized. Thus, Eq. (5) can guarantee that our training and inference targets are the same.

To enhance $h_q^\intercal h_D(d^+)$ approaching $\text{sim}(q, d^+)$ such that Eq. (5) is closer to Eq. (3), we also add the following self-contrastive loss for positive documents as a regularization term:

$$\mathcal{L}_{reg}(q, d^+) = -\log \frac{e^{h_q^\intercal h_D(d^+)}}{\sum_{h \in E_D(d^+)} e^{h_q^\intercal h}}. \quad (6)$$

Thus, our final loss function is

$$\mathcal{L}_{con}(q, d^+, d_1^-, \ldots, d_n^-) + \lambda \mathcal{L}_{reg}(q, d^+), \quad (7)$$

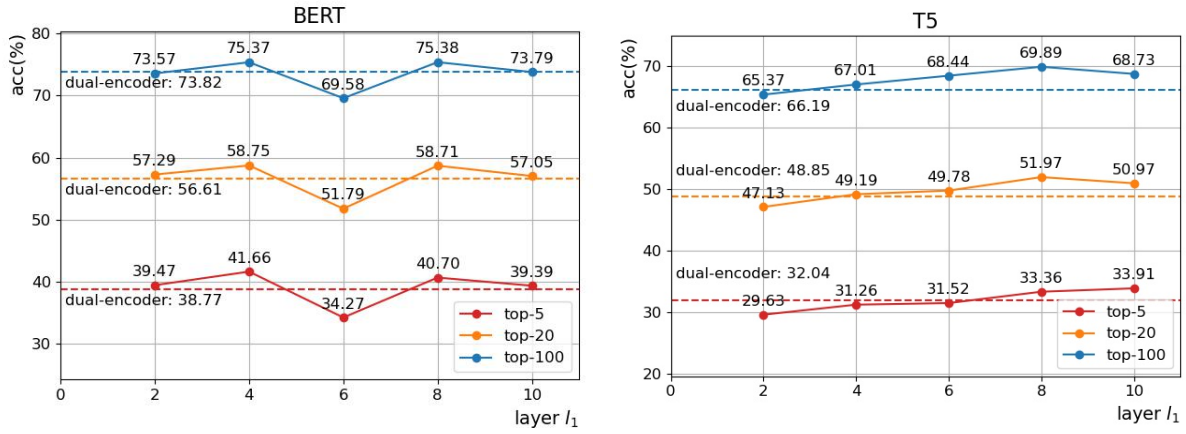

Figure 2: Multi-vector retrieval accuracy w.r.t. different choices of l_1 in a 2-vector MLR model ($\mathcal{S} = \{l_1, l_2 = 12\}$). Results are evaluated on the SQuAD test set with BERT or T5.

	BERT			T5		
Layer Combinations	Top-5	Top-20	Top-100	Top-5	Top-20	Top-100
$\mathcal{S}_{\text{last}} = \{9, 10, 11, 12\}$	41.63	58.68	74.72	31.99	49.51	68.76
$\mathcal{S}_{\text{first}} = \{1, 2, 3, 12\}$	38.54	55.90	72.87	27.28	44.30	63.60
$\mathcal{S}_{\text{uniform}} = \{3, 6, 9, 12\}$	34.62	52.39	70.11	32.88	51.04	69.15

Table 1: Impact of different layer combinations evaluated with a 4-vector MLR model on the SQuAD test set.

where λ is a hyperparameter controlling the strength of the regularization.

Besides, we also experiment with the following two simple pooling strategies: (i) **average pooling**, where we take the average vector of all the vectors in $E_D(d)$, and use it during training and inference as a single vector model; and (ii) **scalar mix pooling**, where we take the weighted average of all the vectors in $E_D(d)$:

$$h_D(d) = \sum_{h_0^{(l)} \in E_D(d)} \text{softmax}(\alpha)_l h_0^{(l)}, \quad (8)$$

with $\alpha \in \mathbb{R}^L$ being a set of learnable parameters for each layer. Similar methods are used to aggregate representations in ELMo (Peters et al., 2018).

3.3 T5 encoders

For the dual-encoder architecture, since T5 does not have a *[CLS]* token whose representations can be used as a document embedding, we follow Ni et al. (2022) to leverage the average of all the token representations in the last layer of a T5 encoder as the document embedding . For our Multi-layer Representations architecture, we adopt the average token vectors in each selected layer as the multi-vector representations of a document.

4 Experiments

4.1 Experimental setup

Datasets and metrics. In our experiments, we adopt the Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and SQuAD (Rajpurkar et al., 2016) datasets processed by Karpukhin et al. (2020). In each of the three datasets, we train on the training questions each of which is attached with one positive and one negative passage sampled from a pool of ∼100 BM25 negatives; we use the dev set for validation and report the top-5/20/100 accuracy on the test set. Top-k accuracy is defined as the fraction of questions whose positive passages appear in the top-k retrieved passages by the model. The number of questions in each dataset is shown in Table 6 in Appendix A. The document collection consists of 21,015,324 Wikipedia passages, which are disjoint text blocks of 100 words.

Training and implementation details. We adopt `bert-base-uncased` or `google-t5/t5-base` from Huggingface (Wolf et al., 2020) as our initial encoder, both of which consist of $L = 12$ transformer layers. Training hyperparameters roughly follow those in Karpukhin et al. (2020), where we use AdamW (Loshchilov and Hutter, 2019) to train our models with an initial learning

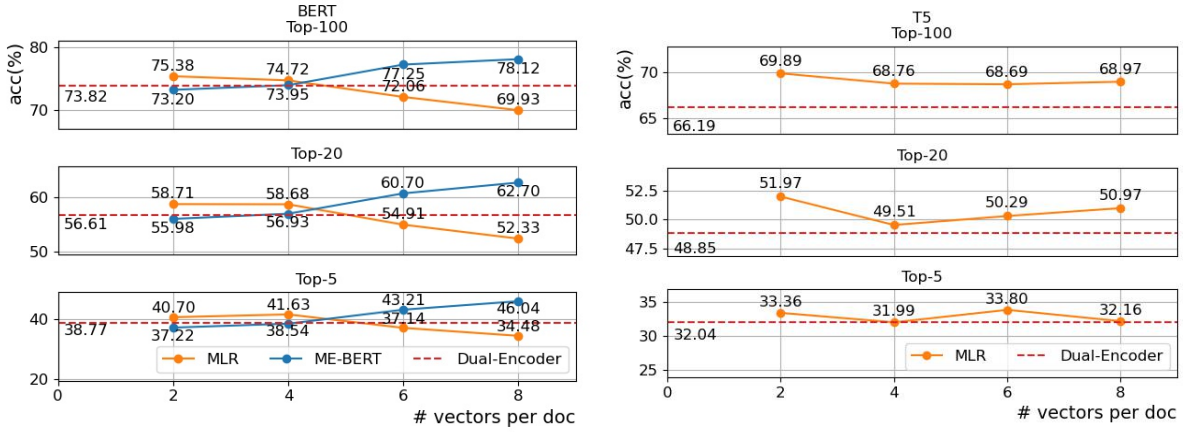

Figure 3: Multi-vector retrieval accuracy w.r.t. the number of vectors per document (i.e., m). Results are evaluated on the SQuAD test set with BERT or T5.

rate of 2e-5 for BERT (5e-4 for T5) and batch size of 128 for 40 epochs. In self-contrastive pooling, we search the regularization strength λ in Eq. (6) from $\{0.01, 0.1, 1, 10\}$. Further details are in Appendix A. We adopt the gradient caching technique (Gao et al., 2021) to use large batch sizes with restricted GPU memories. Our experiments are run on either four Quadro RTX 6000 or four Tesla V100 GPUs. During inference, we first divide the passage collection into 15 shards and encode each of them into a vector file; we then search an `IndexFlatIP` index built from one of the 15 shards so that it can be fed into the memory of four GPUs; we adopt a heap to keep the top results from each shard and merge them to get the final retrieval results.

4.2 Multi-vector retrieval

4.2.1 The impact of layer combinations

We first study a 2-vector MLR model where each document is represented by two vectors from layers $\mathcal{S} = \{l_1, l_2 = 12\}$ in Eq. (1). Results with different choices of l_1 are shown in Figure 2. We can see that, due to different network architectures and training objectives, BERT and T5 exhibit different performance patterns w.r.t. layer combinations. However, for both BERT and T5, the last few layers ($l_1 = 8, 10$) consistently outperform the dual-encoder baseline.

Next, we focus on a 4-vector MLR model, and test three kinds of layer combinations: last four layers where $\mathcal{S}_{\text{last}} = \{9, 10, 11, 12\}$, first three layers plus the last layer where $\mathcal{S}_{\text{first}} = \{1, 2, 3, 12\}$, and uniformly distributed layers where $\mathcal{S}_{\text{uniform}} = \{3, 6, 9, 12\}$. The results are shown in Table 1. We can see that the last few layers perform the best with BERT and comparable to uniformly distributed layers with T5.[3] Therefore, we conclude that the last few layers perform better and more robustly in Multi-layer Representations.

4.2.2 The impact of vector numbers

We next investigate MLR's performance with different number of representation vectors per document, i.e., m in Eq. (1). Specifically, we set $m = 2, 4, 6, 8$, and for each m, we adopt the best performing layer combination according to the analysis in § 4.2.1, i.e., for $m = 2$, we adopt $l_1 = 8$, and for $m = 4, 6, 8$, we adopt the last few layers. We compare MLR with its "last-layer" counterpart, ME-BERT (Luan et al., 2021).[4] Both of the two multi-vector models adopt a fixed number of m vectors to represent a document.

The results are shown in Figure 3. We can see that ME-BERT's performance increases with the increase of vector number m, which means that its representational capacity is increased as expected. For MLR, however, there is a decreasing trend in retrieval performance for both BERT and T5. This indicates that, unlike ME-BERT, MLR's representational capacity cannot scale up well with more representation vectors. This may be attributed to the fact that, Transformer representations in differ-

[3] We further test $m = 6$ with T5, and confirm that the performance of last few layers (Top5/20/100 acc = 33.80/50.29/68.69) is more stable than that of uniformly distributed layers (Top5/20/100 acc = 26.71/44.03/63.03).

[4] We also adapt ME-BERT for T5, but find that neither taking the first m vectors nor taking the average vector plus the first $m - 1$ vectors could outperform the dual-encoder baseline. Therefore we omit ME-BERT's performance with T5 in Figure 3 and leave it for future work.

Models	Natural Questions			TriviaQA			SQuAD		
	Top-5	Top-20	Top-100	Top-5	Top-20	Top-100	Top-5	Top-20	Top-100
Dual-Encoder	68.06	79.34	85.90	71.12	79.63	84.96	38.77	56.61	73.82
Multi-layer Representations (MLR)									
Self-Contrastive ($\mathcal{S}=\{1,\ldots,12\}$)	68.67	79.56	86.26	**71.43**	79.55	85.03	41.00	58.08	74.47
Self-Contrastive ($\mathcal{S}=\{10,12\}$)	**68.73**	**80.00**	86.20	71.40	79.63	**85.28**	**42.47**	**59.93**	**75.79**
ME-BERT									
Average Pooling	68.28	79.22	**86.40**	71.12	79.28	84.96	41.33	58.94	75.36
Scalar Mix Pooling	68.61	79.42	86.23	71.08	79.55	85.03	41.60	59.01	75.57
Self-Contrastive	67.81	78.98	85.90	71.32	**79.70**	85.18	39.02	56.96	73.60
ColBERT									
Average Pooling	68.03	79.22	85.82	71.02	79.40	85.05	39.74	57.37	73.82
Self-Contrastive	68.03	79.36	86.18	71.31	79.69	85.11	38.60	56.08	73.13

Table 2: Single-vector retrieval results on the Natural Questions, TriviaQA, and SQuAD's test sets. We compare the performance of applying the pooling strategies to different multi-vector models (i.e., MLR, ME-BERT, and ColBERT). Best and second best results are in **bold** and underlined, respectively.

ent layers at the same position are more correlated than those in the same layer but at different positions, and therefore adding too many vectors in MLR will limit vector diversity.

On the other hand, we notice that MLR can achieve superior performance over ME-BERT and dual-encoders with a relatively small number of vectors (e.g., $m = 2$ or 4) for both BERT and T5. We further examine which layer is selected on SQuAD's dev set for MLR with $m = 2$ and 4, and find that the last layer is always selected for all the queries. This means a last-layer query representation enhances a bias towards selecting the last-layer document representation, and the performance gains of MLR may largely come from the regularization effect of the max-aggregation during multi-vector training. This observation motivates us to further investigate whether pooling multiple vectors to a single vector through self-contrast could keep these performance gains.

4.3 Single-vector retrieval

In this section, we investigate the single-vector retrieval setting. Specifically, we adopt self-contrastive pooling for Multi-layer Representations, and experiment with pooling from all the layers ($\mathcal{S}=\{1,\ldots,12\}$) and from $m = 2$ layers ($\mathcal{S}=\{10,12\}$) in BERT.[5]

4.3.1 Retrieval results

Single-vector retrieval results of Multi-layer Representations ($\mathcal{S} = \{1,\ldots,12\}$ & $\mathcal{S} = \{10,12\}$), as well as those of ME-BERT (Luan et al., 2021) and ColBERT (Khattab and Zaharia, 2020)[6], are shown in Table 2. For single-vector ME-BERT, we adopt $m = 8$, since this is the best performing setting for multi-vector ME-BERT in Figure 3; for single-vector ColBERT, scalar mix pooling is not applicable, since ColBERT adopts a non-fixed number of vectors to represent each document.

First, on all three datasets, MLR with self-contrastive pooling can usually improve the retrieval accuracy over the dual-encoder baseline by a large margin. For example, on top-5 accuracy, self-contrastive pooling with $\mathcal{S} = \{10,12\}$ leads to a $+3.70\%$ improvement on SQuAD, and a $+0.67\%$ improvement on Natural Questions, while self-contrastive pooling with $\mathcal{S}=\{1,\ldots,12\}$ can lead to a $+0.31\%$ improvement on TriviaQA. Notably, these improvements are made with the same inference time and space complexity as a dual-encoder model.

Second, compared to ME-BERT and ColBERT, MLR can achieve the best performance in most scenarios, which indicates that pooling representations from different layers is more effective than pooling those only in the last layer.

Note also that, compared to other training techniques like retrieval-oriented pre-training or hard negative mining, our single-vector MLR is simple and does not require additional complicated training stages. On the one hand, this means that when we restrict our total computation budget to only one training stage (this is usually desirable for training large-language-model-sized retrievers like repLLaMA (Ma et al., 2024)), our method is still applicable, but the above two methods are not; on the other hand, when we have the budget to do more training stages, our method can be directly

[5] Ablation studies on layer combinations and pooling strategies are in Appendix C.

[6] Different from the original ColBERT, here we only adopt the *[CLS]* embedding for the query to avoid searching multiple times. This is to simplify the training and inference process for efficiency and consistency with other baselines.

	Natural Questions			TriviaQA			SQuAD		
	Top-5	Top-20	Top-100	Top-5	Top-20	Top-100	Top-5	Top-20	Top-100
BM25	–	59.1	73.7	–	66.9	76.7	–	68.8	80.0
GAR	60.9	74.4	85.3	73.1	80.4	85.7	–	–	–
DPR†	68.1	79.3	85.9	71.1	79.6	85.0	38.8	56.6	73.8
ANCE	–	81.9	87.5	–	80.3	85.3	–	–	–
RocketQA	74.0	82.7	88.5	–	–	–	–	–	–
DPR-PAQ	74.5	83.7	88.6	–	–	–	–	–	–
Condenser	–	83.2	88.4	–	81.9	86.2	–	–	–
coCondenser	75.8	84.3	89.0	76.8	83.2	87.3	–	–	–
RetroMAE†	75.5±0.3	84.9±0.2	89.6±0.1	77.3±0.1	**83.7**±0.1	**87.6**±0.0	63.0±0.5	77.2±0.5	86.8±0.2
single-vector MLR	**76.1**±0.4*	**85.0**±0.1	**89.7**±0.1	**77.4**±0.1*	**83.7**±0.1	**87.6**±0.1	**64.0**±0.5*	**77.7**±0.4	**86.9**±0.1

Table 3: Single-vector retrieval results on the Natural Questions, TriviaQA, and SQuAD test sets for various baselines and models trained with retrieval-oriented pre-training and hard negative mining. †: results are reproduced. For RetroMAE and single-vector MLR, we report the mean and standard deviation from five runs with different random seeds. $*$ indicates the improvements of single-vector MLR over RetroMAE is statistically significant ($p < 0.05$). Best results are in **bold**, and "–" means the results are unavailable in the original paper.

integrated with retrieval-oriented pre-training and hard negative mining, which is illustrated in § 4.3.2 below.

4.3.2 Integrating with retrieval-oriented pre-training and hard negative mining

In this section, we integrate MLR in the single-vector retrieval setting with two advanced training techniques: retrieval-oriented pre-training and hard negative mining. Specifically, we adopt a MLR model with $\mathcal{S} = \{10, 12\}$ and self-contrastive pooling. We follow Gao and Callan (2022)'s two-stage training procedure: in the first stage, the model is trained with BM25 negatives; then the trained model is used to mine hard negatives (i.e., top retrieved passages by the trained model in the first stage that do not contain the answer) for questions in the training set; in the second stage, the model is trained with the concatenation of the original and the mined training set. Models in both stages are initialized with a retrieval-oriented pre-trained checkpoint RetroMAE (Xiao et al., 2022). More training details are in Appendix D.

For baselines, we compare to popular sparse (BM25 and GAR (Mao et al., 2021)) and dense (DPR (Karpukhin et al., 2020), ANCE (Xiong et al., 2021), RocketQA (Qu et al., 2021), DPR-PAQ (Oguz et al., 2022), Condenser (Gao and Callan, 2021) and coCondenser (Gao and Callan, 2022)) retrieval systems.

From Table 3, we can see that our single-vector MLR can benefit the baseline dual encoders in most cases. For example, when compared to RetroMAE regarding top-5 accuracy, it leads to a $+0.6\%$ and $+1.0\%$ improvement on Natural Questions and SQuAD, respectively.

	MS MARCO Dev	
	MRR@10	Recall@1000
ANCE	33.0	95.9
SEED	33.9	96.1
coCondenser	38.2	98.4
Aggretriver	36.3	97.3
SPLADE-max	34.0	96.5
SimLM (stage 2)	39.1	98.6
RetroMAE (stage 2)	39.3	98.5
single-vector MLR (stage 1)	37.6	98.5
single-vector MLR (stage 2)	**39.5**	**98.7**

Table 4: Single-vector retrieval results on the MS-MARCO dev set. Best results are in **bold**.

We further evaluate our method on the MS MARCO passage ranking dataset (Nguyen et al., 2016), which contains 502,939 training queries. We report MRR@10 and Recall@1000 on its 6,980 dev queries (MS MARCO Dev), as well as NDCG@10 on the 43 test queries of the TREC 2019 Deep Learning Track (DL'19) (Craswell et al., 2020). The size of the passage collection is 8,841,823. Although distillation from cross encoders can usually achieve superior performance (Ren et al., 2021; Zhang et al., 2022a; Tao et al., 2024), it is computationally much more expensive to train additional cross encoders and generate teacher scores for a large training set. Therefore, we focus on lightweight systems without knowledge distillation, and follow the same two-stage procedure in § 4.3.2 to train MLR. Training hyperparameters are the same as those in Appendix A, except that we adopt an initial learning rate of $1e-5$, total training epochs of 4, and 15 negatives per query for both stages 1 and 2. In stage 2, for each query, we take 100 negatives from the mined hard negatives to make up the negative pool. We adopt

the last checkpoint for inference. λ is set to 0.01 for self-contrastive pooling. For baselines, we compare to ANCE, SEED (Lu et al., 2021), coCondenser, Aggretriver (Lin et al., 2023), SPLADE-max (Formal et al., 2021), SimLM (stage 2) (Wang et al., 2023) and RetroMAE (stage 2) (Xiao et al., 2022).

Results on the MS MARCO dev set are in Table 4, where we can see that single-vector MLR outperforms RetroMAE by 0.2% on both MRR@10 and Recall@1000. On DL'19 test set, single-vector MLR (stage 2) achieves an NDCG@10 of 68.8% compared to RetroMAE's 69.9%. We further find that, if we increase the regularization coefficient λ from 0.01 to 1, single-vector MLR could achieve an NDCG@10 of 70.3%, but with a decrease in MRR@10. Since the NDCG@10 metric emphasizes the first results that the users will see (Craswell et al., 2020), we may adjust the regularization strength to fit different requirements.

4.3.3 Out-of-domain evaluation on BEIR

We additionally conduct out-of-domain evaluation on the BEIR (Thakur et al., 2021) benchmark. We initialize our model with RetroMAE's pre-trained checkpoint, and train the model on 502,939 MS MARCO (Nguyen et al., 2016) training queries, adopting an initial learning rate of $3e-5$, total training epochs of 10, and a λ of 0.01. In the BEIR benchmark, Signal 1M, Arguana, and Quora are considered to be symmetric tasks, where queries and documents are of about the same length and have the same amount of content. Since MLR is trained in an asymmetric manner (i.e., we encode documents and queries in different ways), here we evaluate on the other asymmetric datasets in BEIR. Results are shown in Table 5. We can see that MLR performs significantly better than the dual encoder-based RetroMAE on five datasets (by greater than 1% in NDCG@10), while performs similarly with RetroMAE on the rest. This leads to a $+0.7\%$ improvements in average NDCG@10.

	RetroMAE	single-vector MLR
NQ	0.508	0.515
HotpotQA	0.627	0.620
FiQA-2018	0.300	0.314
TREC-NEWS	0.421	0.406
Robust04	0.432	0.430
Touche-2020	0.257	0.299
CQADupStack	0.309	0.309
DBPedia	0.390	0.390
SCIDOCS	0.151	0.150
FEVER	0.746	0.766
Climate-FEVER	0.220	0.253
SciFact	0.639	0.651
TREC-COVID	0.767	0.764
NFCorpus	0.307	0.305
BioASQ	0.414	0.406
Average	0.432	0.439

Table 5: NDCG@10 results on BEIR for the dual encoder-based RetroMAE and our single-vector MLR.

5 Conclusions

In this study, we proposed to leverage representations from different encoder layers to represent a document in text retrieval. For multi-vector retrieval, we investigated how representations in different layers affect MLR's performance. For single-vector retrieval, we found that MLR can outperform single-vector dual encoder by a large margin, and that pooling representations from different layers is more effective than pooling from representations only in the last layer. We also showed that advanced training techniques such as retrieval-oriented pre-training and hard negative mining can further boost MLR's performance.

Limitations

Our current method is asymmetric in terms of representing queries and documents, i.e., we represent each document with multiple vectors and regularize them with the self-contrastive pooling loss, while only representing each query with a single vector. This is for efficiency consideration, but may limit our method's capacity and decrease its performance on symmetric search problems such as duplicate question identification and argument mining. It would be a direct future work to explore utilizing multiple vectors to represent the query as well. Besides, the improved retrieval system may have broader societal impacts such as insufficient representations of minority groups, exhibiting stereotyped or biased results, and so on. Detection and mitigation of such unintended behaviors is of great significance but beyond the scope of this paper.

Acknowledgments

This work was supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, by the AXA Research Fund, and by the EU TAILOR grant 952215. We also acknowledge the use of Oxford's ARC facility, of the EPSRC-funded Tier 2 facilities JADE (EP/P020275/1), and of GPU computing support by Scan Computers International Ltd.

References

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 deep learning track. *Preprint*, arXiv:2003.07820.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Philipp Ennen, Federica Freddi, Chyi-Jiunn Lin, Po-Nien Kung, RenChu Wang, Chien-Yi Yang, Dashan Shiu, and Alberto Bernacchia. 2023. Hierarchical representations in dense passage retrieval for question-answering. In *Proceedings of the Sixth Fact Extraction and VERification Workshop (FEVER)*, pages 17–28, Dubrovnik, Croatia. Association for Computational Linguistics.

Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. 2022. Head2Toe: Utilizing intermediate representations for better transfer learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6009–6033. PMLR.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *Preprint*, arXiv:2109.10086.

G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 316–321, Online. Association for Computational Linguistics.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4143–4152, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

MohammadSaleh Hosseini, Munawara Munia, and Latifur Khan. 2023. BERT has more to offer: BERT layers combination yields better sentence embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15419–15431, Singapore. Association for Computational Linguistics.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized

late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. 2023. Aggretriever: A simple approach to aggregate textual representations for robust dense passage retrieval. *Transactions of the Association for Computational Linguistics*, 11:436–452.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Zheng Liu, Shitao Xiao, Yingxia Shao, and Zhao Cao. 2023. RetroMAE-2: Duplex masked auto-encoder for pre-training retrieval-oriented language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2635–2648, Toronto, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong Siamese encoder for dense text retrieval using a weak decoder. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2791, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning LLaMA for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2421–2425, New York, NY, USA. Association for Computing Machinery.

Nikolaos Manginas, Ilias Chalkidis, and Prodromos Malakasiotis. 2020. Layer-wise guided training for BERT: Learning incrementally refined document representations. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 53–61, Online. Association for Computational Linguistics.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100, Online. Association for Computational Linguistics.

Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15376–15400. PMLR.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: a human generated machine reading comprehension dataset.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. 2018. Multi-level abstraction convolutional model with weak supervision for information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 985–988, New York, NY, USA. Association for Computing Machinery.

Barlas Oguz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Scott Yih, Sonal Gupta, and Yashar Mehdad. 2022. Domain-matched pre-training tasks for dense retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1524–1534, Seattle, United States. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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*, pages 2825–2835, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Si Sun, Chenyan Xiong, Yue Yu, Arnold Overwijk, Zhiyuan Liu, and Jie Bao. 2022. Reduce catastrophic forgetting of dense retrieval training with teleportation negatives. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6639–6654, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yun Tang, Juan Pino, Xian Li, Changhan Wang, and Dmitriy Genzel. 2021. Improving speech translation by understanding and learning from the auxiliary text translation task. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4252–4261, Online. Association for Computational Linguistics.

Chongyang Tao, Chang Liu, Tao Shen, Can Xu, Xiubo Geng, Binxing Jiao, and Daxin Jiang. 2024. ADAM: Dense retrieval distillation with adaptive dark examples. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11639–11651, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

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)*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. SimLM: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2244–2258, Toronto, Canada. Association for Computational Linguistics.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5075–5084. PMLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Bohong Wu, Zhuosheng Zhang, Jinyuan Wang, and Hai Zhao. 2022. Sentence-aware contrastive learning for open-domain passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1062–1074, Dublin, Ireland. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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*.

Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. 2015. HD-CNN: Hierarchical deep convolutional neural networks for large scale visual recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2740–2748.

Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2022a. Adversarial retriever-ranker for dense text retrieval. In *International Conference on Learning Representations*.

Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022b. Multi-view document representation learning for open-domain dense retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5990–6000, Dublin, Ireland. Association for Computational Linguistics.

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K. Qiu, and Lili Qiu. 2024. Retrieval augmented generation (RAG) and beyond: A comprehensive survey on how to make your llms use external data more wisely. *Preprint*, arXiv:2409.14924.

A Additional Training Details

We adopt two separate BERT (or T5 encoder) from Huggingface (Wolf et al., 2020) as our query and document encoder. Statistics of the adopted datasets are shown in Table 6. For all the experiments, we adopt a linear learning rate scheduler with warm up proportion 0.05. We clip the gradient if it is larger than 2.0. Maximum input sequence length for BERT (or T5 encoder) is set to 256. For the validation metric, we follow Karpukhin et al. (2020) to initially adopt cross entropy loss and then change to average rank on the dev set after 30 epochs. For Natural Questions and SQuAD, the best dev checkpoint is adopted for inference, while for TriviaQA, we adopt the last checkpoint. In self-contrastive pooling, we search the regularization strength λ in Eq. (6) from $\{0.01, 0.1, 1, 10\}$, and adopt $\lambda = 1$ for MLR ($\mathcal{S} = \{10, 12\}$) and ME-BERT, $\lambda = 0.1$ for MLR ($\mathcal{S} = \{1, \ldots, 12\}$), and $\lambda = 0.01$ for ColBERT. It typically takes 16 to 30 hours to train one model on one dataset and encode the passage collection with four V100 GPUs, and 60 to 500GB disk space to store the encoded vector files. Due to long training time and large storage requirements, all the results in this paper (except those in § 4.3.2) are from a single run using seed 12345, which is consistent with Karpukhin et al. (2020) and most other works. We adhere to the licenses and intended use of the pre-trained checkpoints and datasets provided in their original papers or on their websites.

B Additional Multi-vector Retrieval Results on Natural Questions

Additional multi-vector retrieval results on Natural Questions with BERT are provided in Figure 4 and 5, which show similar trends to those on SQuAD with BERT (the left subgraph of Figure 2 and 3).

Dataset	Train (original / filtered)	Dev	Test
NQ	79,168 / 58,880	8,757	3,610
TriviaQA	78,785 / 60,413	8,837	11,313
SQuAD	78,713 / 70,096	8,886	10,570

Table 6: Number of questions in each dataset. For the training questions, we follow Karpukhin et al. (2020) to filter out questions with no associated positive passages, and the number of the remaining questions are presented after the slash. NQ stands for Natural Questions.

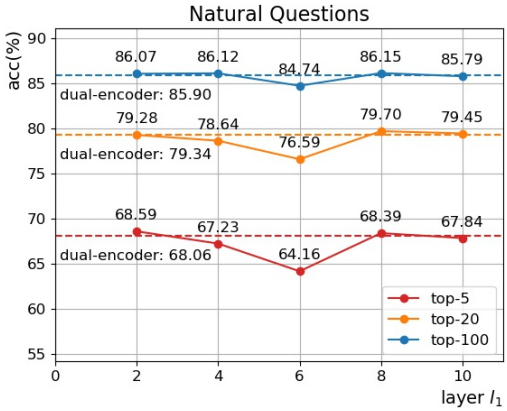

Figure 4: Multi-vector retrieval accuracy w.r.t. different choices of l_1 in a 2-vector MLR model ($\mathcal{S} = \{l_1, l_2 = 12\}$). Results are evaluated on the Natural Questions test set with BERT.

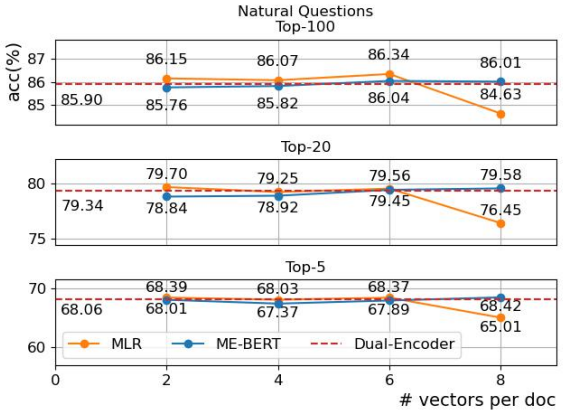

Figure 5: Multi-vector retrieval accuracy w.r.t. the number of vectors per document (i.e., m). Results are evaluated on Natural Questions test set with BERT.

C Ablation Studies on Pooling Strategies and Layer Combinations for Single-vector MLR

We first compare different pooling strategies used in MLR. The results on Natural Questions are shown in Table 7, where we can see that self-contrastive pooling consistently performs better than average pooling and scalar mix pooling.

Next, we study the performance of different layer combinations in the 2-vector MLR model with self-contrastive pooling. The results on Natural Questions are shown in Table 8, where we can see that pooling from layers $\mathcal{S} = \{10, 12\}$ achieves the best top-5 and top-20 accuracy, while obtains a reasonable top-100 accuracy. Therefore we adopt $\mathcal{S} = \{10, 12\}$ in our experiment.

Pooling Methods	Natural Questions Top-5	Top-20	Top-100
MLR ($\mathcal{S}=\{1,\ldots,12\}$)			
Average	67.17	78.34	85.60
Scalar Mix	66.81	78.59	85.43
Self-Contrastive	**68.67**	**79.56**	**86.26**
MLR ($\mathcal{S}=\{10,12\}$)			
Average Pooling	68.03	79.42	85.93
Scalar Mix Pooling	68.31	79.78	85.87
Self-Contrastive	**68.73**	**80.00**	**86.20**

Table 7: Single-vector retrieval accuracy w.r.t. different pooling methods for MLR with layer combinations $\mathcal{S}=\{1,\ldots,12\}$ and $\mathcal{S}=\{10,12\}$. Best results for each layer combination are in **bold**.

	Natural Questions Top-5	Top-20	Top-100
$\mathcal{S}=\{2,12\}$	67.98	79.47	85.96
$\mathcal{S}=\{4,12\}$	67.78	<u>79.86</u>	86.09
$\mathcal{S}=\{6,12\}$	<u>68.34</u>	79.50	**86.26**
$\mathcal{S}=\{8,12\}$	67.42	78.73	86.01
$\mathcal{S}=\{10,12\}$	**68.73**	**80.00**	<u>86.20</u>

Table 8: Single-vector retrieval accuracy w.r.t. different layer combinations for a 2-vector MLR model with self-contrastive pooling. Best and second best results are in **bold** and <u>underlined</u>, respectively.

D Training Details for Experiments with Retrieval-oriented Pre-training and Hard Negative Mining

Training hyperparameters are the same as those in Appendix A, except that, for stage 1, we adopt an initial learning rate of $5e-6$ and total training epochs of 40; for stage 2, an initial learning rate of $5e-6$ and total training epochs of 20. In stage 2, for each question, we take 50 negatives from each of the original and generated training set to make up a negative pool of 100 negatives. We adopt the last checkpoint for inference on Natural Questions and TriviaQA following Gao and Callan (2022), and the best dev checkpoint for SQuAD. λ is set to 0.1 for self-contrastive pooling.