Curriculum Learning for Dense Retrieval Distillation

Hansi Zeng
University of Massachusetts Amherst
United States
hzeng@cs.umass.edu

Hamed Zamani
University of Massachusetts Amherst
United States
zamani@cs.umass.edu

Vishwa Vinay
Adobe Research
India
vinay@adobe.com

ABSTRACT

Recent work has shown that more effective dense retrieval models can be obtained by distilling ranking knowledge from an existing base re-ranking model. In this paper, we propose a generic *curriculum learning* based optimization framework called CL-DRD that controls the difficulty level of training data produced by the re-ranking (teacher) model. CL-DRD iteratively optimizes the dense retrieval (student) model by increasing the difficulty of the knowledge distillation data made available to it. In more detail, we initially provide the student model coarse-grained preference pairs between documents in the teacher's ranking, and progressively move towards finer-grained pairwise document ordering requirements. In our experiments, we apply a simple implementation of the CL-DRD framework to enhance two state-of-the-art dense retrieval models. Experiments on three public passage retrieval datasets demonstrate the effectiveness of our proposed framework.

CCS CONCEPTS

• **Information systems** → **Document representation**; **Learning to rank**;

KEYWORDS

Neural Ranking Models; Dense Retrieval; Knowledge Distillation; Curriculum Learning

ACM Reference Format:
Hansi Zeng, Hamed Zamani, and Vishwa Vinay. 2022. Curriculum Learning for Dense Retrieval Distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531791

1 INTRODUCTION

Retrieval that combines high dimensional vector representations of queries and documents obtained from deep neural networks and approximate nearest neighbor search algorithms have recently attracted considerable attention [12, 23, 32]. These dense retrieval models rely on the availability of large-scale training data, which includes public datasets such as MS MARCO [3], and proprietary datasets collected from the query logs of deployed search engines. While the numbers of queries and documents are quite large, the datasets often suffer from incomplete relevance judgments [23], i.e., very few documents are judged for a given query. An approach to address this sparsity issue is to train dense retrieval models using *knowledge distillation*. Recent work [9–11, 16, 26] has shown that the performance of dense retrieval models (i.e., *the student models*) can be improved by distilling ranking knowledge from a more expensive re-ranking model (i.e., *the teacher model*) that learns representations based on the interactions between query and document terms using cross-encoders [21, 24].

In the knowledge distillation setting, an available teacher model assigns a distinct score to a query-document pair on which the supervision signal for optimizing the dense retrieval student model is based. Since the teacher can effectively score all pairs of queries and documents, we are not limited by the availability of labeled data, thereby providing us with greater flexibility. In this paper, we take advantage of this flexibility and introduce a generic *curriculum learning* framework for training dense retrieval models via knowledge distillation. The core idea of the curriculum learning (CL) is to provide a systematic approach to decompose the complex knowledge and design a curriculum for learning concepts from simple to hard [8, 15, 31]. Motivated by curriculum learning's ability to find better local optima [1], we propose a framework called CL-DRD that introduces an iterative optimization process in which the difficulty level of the training data produced using the teacher model, as made available to the student, increases at every iteration. Through this CL-DRD process, we first demand the dense retrieval student model to recover coarse-grained distinctions between the documents exampled by the teacher model and then progressively move towards recovering finer-grained ordering of documents. For robust iterative optimization of the dense retrieval models, we adapt the listwise loss function of LambdaRank [2] to our knowledge distillation setting. Therefore, our loss function only focuses on the order of documents produced by the teacher model, and not the exact document scores.

In our experiments, we apply a simple implementation of the proposed optimization framework to two state-of-the-art dense retrieval models. First, we enhance TAS-B [11], a model that uses a single representation vector for each query and document. Second, we repeat our experiments with the ColBERTv2 model [30], a recent dense retrieval model that uses multiple representations per query and document. Our experiments on three public passage retrieval benchmarks demonstrate the effectiveness of the proposed framework. To improve the reproducibility of our models, we release the source code and the parameters of our models for research purposes.[1]

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531791

[1] https://github.com/HansiZeng/CL-DRD

2 METHODOLOGY

2.1 Background

2.1.1 Dense Retrieval. This paper focuses on the task of retrieving items based on high-dimensional dense learned representations for queries and documents, which is often called *dense retrieval*. The query and document representations in dense retrieval models are often obtained using large-scale pre-trained language models, such as BERT [7], fine-tuned for the downstream retrieval task [12, 17, 32]. Dense retrieval models can be seen as a category of vector space models [28]. Dense retrieval models compute the relevance score for a query q and a document d as follows:

$$\text{score}(q, d) = \text{sim}(E_\psi(q), E_\phi(d))$$

where $E_\psi()$ and $E_\phi()$ are the query encoders parameterized by ψ and the document encoder parameterized by ϕ, respectively. The encoders produce a dense representation of the given input. They often share the same output dimensionality: $|E_\psi(q)| = |E_\phi(d)|$. The similarity function 'sim' is often implemented using the inner product. For efficient retrieval, dense retrieval models often employ approximate nearest neighbor (ANN) search algorithms.

2.1.2 Knowledge Distillation. For optimizing dense retrieval models, a ranking loss function is employed. The loss function is often based on either pointwise, pairwise, or listwise modeling, similar to learning-to-rank models. Since dense retrieval models often consist of millions or billions of trainable parameters, existing datasets are not often sufficient for training them. Recent work [11, 16, 26] has successfully employed knowledge distillation (KD) for training dense retrieval models, where a teacher model produces training data for training a student model (i.e., the dense retrieval model). Teacher models are often more complex, with higher capacity and lower efficiency. A common approach is to use a cross encoding neural ranking model as the teacher model. In cross encoding models, both the given query and document are encoded jointly, leading to a superior result compared to dual encoders because of capturing more interaction information between the given query and document [21, 24]. We also use a cross encoding model in our experiments as the teacher. In more detail, we use a BERT model [21] that takes `[CLS] query tokens [SEP] document tokens [SEP]` as input and the relevance score is obtained by the linear transformation of the `[CLS]` representation.

2.2 The CL-DRD Framework

Learning-to-rank models, including neural ranking models, are often trained based on the relevance judgment information. For instance, in pairwise models, a relevant document is paired with a sampled non-relevant document to form a training instance. However, in knowledge distillation, the training instances are produced based on the teacher model's output. This gives us substantial flexibility in producing the training data and we are not limited to the relevant documents that appeared in the relevance judgment file.

We take advantage of this flexibility and introduce the CL-DRD framework. It combines the ideas of curriculum learning and knowledge distillation. **The intuition behind CL-DRD is to introduce an iterative training process in which the difficulty of training data in each iteration increases.** This iterative optimization process is introduced in Algorithm 1. In this algorithm, the parameter δ controls the difficulty of training sets generated by the 'RankingDataGeneration' function. It starts with the difficulty level of 1 and its difficulty increases over time, until a stopping criterion is met. This stopping criterion can be based on the ranking performance on a held-out validation set, or can be based on a constant number of iterations.

Algorithm 1 The Iterative Optimization Process in CL-DRD.

1: **Input** (a) training query set Q; (b) document collection C; (c) *optional* relevance judgment set R; (d) teacher model $\widehat{M}$.
2: **Output** dense retrieval model M_θ.
3: **Initialize** θ.
4: $\delta \leftarrow 1$ ▷ δ denotes the difficulty level in CL data
5: **repeat**
6: $D_\delta \leftarrow \text{RankingDataGeneration}(\delta; \widehat{M}, Q, C, R)$
7: $\theta^* \leftarrow \arg\min_\theta \mathcal{L}_{KD}(M_\theta, D_\delta)$
8: $\delta \leftarrow \delta + 1$
9: **until** stopping criterion is met
10: **return** M_{θ^*}

CL-DRD is a generic framework and can be implemented in many different ways. For example, the difficulty of training sets can be modeled based on different assumptions. In this paper, we provide details of one of these implementations. In the following, we describe how we generate training data at each iteration (Algorithm 1; line 6) and how we optimize the student model (Algorithm 1; line 7). The other experimental details such as initialization (line 3) and the stopping criterion (line 10) are reported in Section 3.

2.2.1 Generating Training Data in Each Iteration for Curriculum Learning. In typical curriculum learning, training instances are sorted or weighted according to their difficulty level and are fed to the model for optimization [1, 18, 19, 22]. Here, we use the same high-level idea, but with a substantially different approach. The 'RankingDataGeneration' function in Algorithm 1 is supposed to produce more difficult ranking training data as δ increases. There are numerous ways to generate training data using knowledge distillation with different difficulty levels. Without loss of generality, this section describes the approach we choose for our experiments. In our approach, the training query set Q remains the same for all iterations.[2] For each training query $q \in Q$, we take the top 200 documents returned by the student dense retrieval model and re-rank them using the teacher model, and then create three groups of documents: (1) the pseudo-relevant group: first K documents returned by the teacher model, (2) the 'hard negative' group: the next K' documents in the ranked list returned by the teacher model, and (3) the remaining K' documents in the ranked list produced by the teacher model. For each query, we keep the number of documents used for the optimization process constant and equal to L. Therefore, for training the student model in every iteration, we select K documents from Group 1 (called $S_q^{(1)}$), we randomly sample N_h documents from Group 2 (called $S_q^{(2)}$), and we finally randomly sampled N_s documents from Group 3 (called $S_q^{(3)}$), where $L = K +$

[2]Iteration refers to the curriculum learning iterations in Algorithm 1. Each iteration consists of many batches of training data.

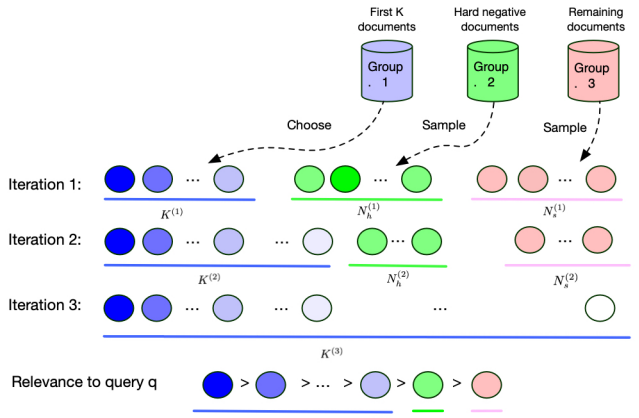

Figure 1: The data creation process in each iteration of curriculum learning based on knowledge distillation.

$N_h + N_s$. In the first iteration, we start with a relatively small value of K and increase this value in subsequent iterations. Naturally, this leads to smaller $N_h + N_s$. Figure 1 provides an illustration of the process.

Therefore, for each query, we produce a ranked list of L documents with the following pseudo-labels:

$$y_q^t(d) = \begin{cases} \frac{1}{r_{qd}^t} & \text{iff } d \in S_q^{(1)} \\ 0 & \text{iff } d \in S_q^{(2)} \\ -1 & \text{iff } d \in S_q^{(3)} \end{cases} \quad (1)$$

where $r_{q,d}^t$ is the ranking position of the document d given the query q in the teacher ranked list. As we present in Section 2.2.2, we use a loss function that only considers the order of document and the exact pseudo-label values do not impact the loss value. We use a listwise loss function that does not rely on the exact document scores produced by the teacher model. To better understand our reasons for using such a pseudo-labeling strategy, let us categorize every document pairs with different pseudo-labels into four document pair types:

- *Type 1 pairs*: any two distinct documents from $S_q^{(1)}$. The document ranked higher by the teacher model is considered more relevant. This results in $\frac{K(K-1)}{2}$ document pairs for training.
- *Type 2 pairs*: any document pair $d \in S_q^{(1)}$ and $d' \in S_q^{(2)}$. The document d is considered more relevant. This results in KN_h document pairs for training.
- *Type 3 pairs*: any document pair $d \in S_q^{(1)}$ and $d' \in S_q^{(3)}$. The document d is considered more relevant. This results in KN_s document pairs for training.
- *Type 4 pairs*: any document pair $d \in S_q^{(2)}$ and $d' \in S_q^{(3)}$. The document d is considered more relevant. This results in $N_h N_s$ document pairs for training.

Learning from all *Type 2, 3, 4* pairs enforces the dense retrieval student model M_θ to distinguish documents from different groups, which is expected to be easier than than distinguishing document pairs in *Type 1*. The reason is that Type 1 pairs enforce the student model to learn the exact ordering of documents provided by the teacher model.

In our experiments, we consider three iterations of curriculum learning. From iteration 1 to 3, the value K increases from 5 to 10 to 30, respectively. In each iteration, the student model is trained for a fixed number of epochs. Since the number of pairs in Type 1 increases from 10 to 45 and to 435 and Type 1 is expected to contain the most difficult document pairs, **the difficulty level of training data in each iteration is expected to increase.** Hence, in our curriculum learning based knowledge distillation algorithm, we progressively require that the student model to concentrate on more fine-grained differences in the output provided by the teacher model.

2.2.2 Optimizing the Student Model through Knowledge Distillation. Inspired by LambdaRank [2], we use the following listwise loss function for training our student model using knowledge distillation at each iteration:

$$\mathcal{L}_{KD}(M_\theta, D_\delta) = \sum_{(q,S_q) \in D_\delta} \sum_{d,d' \in S_q} y_q^t(d,d') w(d,d') \log(1 + e^{M_\theta(q,d') - M_\theta(q,d)}) \quad (2)$$

where $S_q = S_q^{(1)} \cup S_q^{(2)} \cup S_q^{(3)}$ denotes all the documents selected via the pseudo-labeling approach presented in Section 2.2.1, $y_q^t(d,d') = \mathbb{1}\{y_q^t(d) > y_q^t(d')\}$. The function $w(d,d')$ is equal to $|\frac{1}{\pi_q(d)} - \frac{1}{\pi_q(d')}|$, where $\pi_q(d)$ denotes the rank of document d in the result list produced by the student dense retrieval model M_θ.

3 EXPERIMENTS

Datasets: We trained our model in the MS MARCO passage retrieval dataset [3] which contains approximated 8.8M passages and 503K training queries with shallow annotations (≈1.1 relevant passages per query on average). For the model evaluation, we use three datasets: (1) MS MARCO-Dev that contains 6980 labeled queries, (2) TREC-DL'19: the passage retrieval dataset used in the 2019 edition of TREC Deep Learning Track [4] with 43 queries, and (3) TREC-DL'20: the passage retrieval dataset of TREC Deep Learning Track 2020 [5] with 54 queries. For evaluation, we report MAP@1000 for all three datasets, as well as the official metrics MRR@10 for MS MARCO and nDCG@10 for TREC-DL'19 and TREC-DL'20.

Experiment Settings: For the single-vector dense retrieval model, we use the DistilBERT [29] with the pre-trained checkpoint made available from TAS-B [11] as the initialization. For the multi-vector dense retrieval model, we also use the DistilBERT [29] as the backbone. As the re-ranking teacher model, we use the MiniLM cross-encoder that is publicly available on HuggingFace.[3] We use the Adam optimizer [14] with linear scheduling with the warmup of 4000 steps and initial learning rate $[7e^{-6}, 3e^{-6}, 3e^{-6}]$ for the three CL iterations. We set the batch size to 8 and the maximum length for queries and passages to 30 and 256 tokens, respectively. For the three iterations, the sizes of group 1: [5, 10, 30], group 2: [45, 40, 20], group 3: [150, 150, 150]. The number of sampled documents for each

[3] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2

Table 1: The performance comparison with state-of-the-art baselines. We use the two-tailed paired t-test with $p < 0.05$. The superscripts refer to significant improvements compared to all sparse retrieval models (∗), ANCE and ADORE (†), TCT-ColBERT and Margin-MSE (‡), TAS-B (§), and ColBERTv2 (¶). "-" denotes the results that are not applicable or available.

Model	KD	Encoder	#params	MS MARCO DEV		TREC-DL'19		TREC-DL'20	
				MRR@10	MAP@1k	nDCG@10	MAP@1k	nDCG@10	MAP@1k
Sparse Retrieval									
BM25 [27]	-	-	-	.187	.196	.497	.290	.487	.288
DeepCT [6]	-	-	-	.243	.250	.550	.341	.556	.343
docT5query [20]	-	-	-	.272	.281	.642	.403	.619	.407
Multi-Vector Dense Retrieval									
ColBERT [13]	✗	BERT-Base	110M	.360	-	-	-	-	-
ColBERTv2 [30]	✓	BERT-Base	110M	**.397**	-	-	-	-	-
ColBERTv2 [30]	✓	DistilBERT	66M	.384	.389	**.733**	.464	.712	.473
ColBERTv2 + CL-DRD (Ours)	✓	DistilBERT	66M	.394*§	**.398***§¶	.727*§	**.472***§¶	**.717***§	**.487***§¶
Single-Vector Dense Retrieval									
ANCE [32]	✗	BERT-Base	110M	.330	.336	.648	.371	.646	.408
ADORE [33]	✗	BERT-Base	110M	.347	.352	.683	.419	.666	.442
RocketQA [25]	✓	ERNIE-Base	110M	.370	-	-	-	-	-
TCT-ColBERT [16]	✓	BERT-Base	110M	.335	.342	.670	.391	.668	.430
Margin-MSE [10]	✓	DistilBERT	66M	.325	.331	.699	.405	.645	.416
TAS-B [11]	✓	DistilBERT	66M	.344	.351	.717	.447	.685	.455
TAS-B + CL-DRD (Ours)	✓	DistilBERT	66M	**.382***§	**.386***§	**.725***§	**.453***‡	**.687***‡	**.465***§

query: $L = 30$, and $K = [5, 10, 30]$, $N_h = [12, 10, 0]$, $N_s = [13, 10, 0]$ for the three iterations.

The CL-DRD models: CL-DRD is a generic optimization framework for dense retrieval models that can be applied to any dense retrieval model. In our experiments, we trained two different dense retrieval models using CL-DRD: (1) **TAS-B + CL-DRD**: TAS-B is the best performing dense retrieval baseline that uses a *single vector* representation for each query and document. In TAS-B + CL-DRD we apply our framework to the TAS-B model. (2) **ColBERTv2 + CL-DRD**: Similarly, we choose ColBERTv2 the best performing dense retrieval model that uses *multiple vectors* per query and document. Note that this model uses our own implementation of ColBERTv2 that uses DistilBERT. We compare our models with several state-of-the-art baselines shown in the Table 1.

Results and Discussion: The retrieval results obtained by our models and the baselines are reported in Table 1. According to the results, dense retrieval models generally outperform lexical matching models including the ones that use pre-trained language models for document expansion, such as docT5query [20]. TAS-B + CL-DRD outperforms all the dense retrieval baselines that use a single vector for representing each query and document. This improvement is consistent across all three datasets, and is statistically significant in most cases. Note that some of these baseline models use significantly larger models compared to ours. The number of parameters for each model is mentioned in Table 1. Since TAS-B + CL-DRD uses TAS-B [11] as its parameter initialization, we can have a direct comparison with this baseline. Applying CL-DRD to TAS-B leads to 11% improvements on the MS MARCO Dev set in terms of MRR@10.

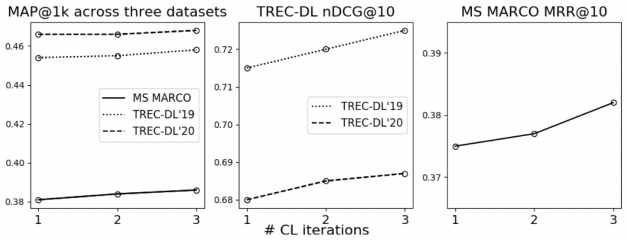

Figure 2: The results obtained by TAS-B + CL-DRD at different iterations of curriculum learning.

Table 1 suggests that CL-DRD can also improve dense retrieval models with multiple vector representations. When we apply CL-DRD to ColBERTv2 (the current state-of-the-art in dense retrieval with DistilBERT), we obtain improvements in terms of all metrics on all collections except for nDCG@10 on TREC DL'19. The improvements are larger for recall-oriented metrics, e.g., MAP@1000, and they are statistically significant in majority of cases. This demonstrates that CL-DRD is sufficiently flexible to be applied to different dense retrieval models and introduce significant improvements.

In general, ColBERTv2 + CL-DRD performs better than TAS-B + CL-DRD. It is worth noting that representing queries and documents with multiple vectors demands significantly higher memory requirements, indexing cost, and query processing cost.

Ablation Study: We conduct a few experiments for evaluating the effectiveness of curriculum learning. For the sake of space, we solely focus on the TAS-B + CL-DRD model. Figure 2 demonstrates the performance of this model after each curriculum learning iteration

on all three datasets. As shown in the figure, the performance generally improves at each iteration. This demonstrates that the proposed CL approach is effective for training ranking models. To make sure that this improvement is not just due to the number of epochs or the size of training sets, we additionally train our model on a reverse curriculum learning setup. In this experiment, we followed the same procedure with the same number of iterations, but we start from the last iteration used in our CL approach. We observe that our model still outperforms this reverse CL approach. For example, this approach achieves an nDCG@10 of 0.715 and 0.683 on TREC DL'19 and TREC DL'20, respectively. It also achieves an MRR@10 of 0.378 on MS MARCO Dev set, which is significantly lower than the results obtained by our method.

4 CONCLUSIONS AND FUTURE WORK

We introduced CL-DRD, a generic framework for optimizing dense retrieval models through knowledge distillation. Inspired by curriculum learning, CL-DRD follows an iterative process in which supervision of increasing levels of difficulty are derived from the teacher model's output. We provided a simple implementation of this framework and demonstrated its effectiveness on three public passage retrieval benchmarks.

In the future, we intend to explore more sophisticated solutions for controlling the difficulty of each iteration in CL-DRD. We are also interested in developing machine learning models that can select informative training instances based on the teacher's performance.

5 ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, and in part by gift funding from Adobe Research. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09.*

[2] Christopher J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview.

[3] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *30th Conference on Neural Information Processing Systems, NIPS* (2016).

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2019. Overview of the TREC 2019 Deep Learning Track. In *TREC*.

[5] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *TREC*.

[6] Zhuyun Dai and Jamie Callan. 2020. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* abs/1910.10687 (2020).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT* (2019).

[8] Jeffrey L. Elman. 1993. Learning and development in neural networks: the importance of starting small. *Cognition* 48 (1993), 71–99.

[9] Luyu Gao and Jamie Callan. 2021. Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. *ArXiv* abs/2108.05540 (2021).

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

[11] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[12] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP* (2020).

[13] O. Khattab and Matei A. Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations, ICLR* (2015).

[15] Kai A. Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition* 110 (2009), 380–394.

[16] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy J. Lin. 2021. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)* (2021).

[17] Y Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics, TACL* (2021).

[18] Sean MacAvaney, Franco Maria Nardini, R. Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Training Curricula for Open Domain Answer Re-Ranking. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[19] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2020. Teacher–Student Curriculum Learning. *IEEE Transactions on Neural Networks and Learning Systems* 31 (2020), 3732–3740.

[20] Rodrigo Nogueira. 2019. From doc2query to docTTTTTquery.

[21] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *ArXiv* abs/1901.04085 (2019).

[22] Gustavo Penha and Claudia Hauff. 2020. Curriculum Learning Strategies for IR: An Empirical Study on Conversation Response Ranking. In *Proceedings of the 42nd European Conference on IR Research (ECIR '20)*. Springer-Verlag, Berlin, Heidelberg, 699–713.

[23] Prafull Prakash, Julian Killingback, and Hamed Zamani. 2021. Learning Robust Dense Retrieval Models from Incomplete Relevance Labels. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[24] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *ArXiv* abs/1904.07531 (2019).

[25] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, 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, NAACL*.

[26] 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. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP* (2021).

[27] Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*. Gaithersburg, MD: NIST, 109–126.

[28] Gerard Salton, A. Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18 (1975), 613–620.

[29] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* abs/1910.01108 (2019).

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

[31] Lilian Weng. 2020. Curriculum for Reinforcement Learning. *lilianweng.github.io/lil-log* (2020). https://lilianweng.github.io/lil-log/2020/01/29/curriculum-for-reinforcement-learning.html

[32] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *9th International Conference on Learning Representations, ICLR* (2021).

[33] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).