CODER: An efficient framework for improving retrieval through COntextual Document Embedding Reranking

George Zerveas[1], **Navid Rekabsaz**[2], **Daniel Cohen**[1,3], **Carsten Eickhoff**[1]

[1]AI Lab, Brown University, USA {george_zerveas,carsten}@brown.edu
[2]Johannes Kepler University Linz, LIT AI Lab, Austria navid.rekabsaz@jku.at
[3]Dataminr, USA daniel.cohen@dataminr.com

Abstract

Contrastive learning has been the dominant approach to training dense retrieval models. In this work, we investigate the impact of *ranking context* – an often overlooked aspect of learning dense retrieval models. In particular, we examine the effect of its constituent parts: jointly scoring a large number of negatives per query, using retrieved (query-specific) instead of random negatives, and a fully list-wise loss. To incorporate these factors into training, we introduce Contextual Document Embedding Reranking (CODER), a highly efficient retrieval framework. When reranking, it incurs only a negligible computational overhead on top of a first-stage method at run time (~ 5 ms delay per query), allowing it to be easily combined with any state-of-the-art dual encoder method. Models trained through CODER can also be used as stand-alone retrievers. Evaluating CODER in a large set of experiments on the MS MARCO and TripClick collections, we show that the contextual reranking of precomputed document embeddings leads to a significant improvement in retrieval performance. This improvement becomes even more pronounced when more relevance information per query is available, shown in the TripClick collection, where we establish new state-of-the-art results by a large margin.

1 Introduction

Neural text retrieval models typically rely on a contrastive training optimization that uses *(query, positive document, negative document)* triplets as training samples. This scheme is especially popular, as it is well-suited to the computational constraints of large transformer-based language models such as BERT (Devlin et al., 2019) and its variants for retrieval (Nogueira and Cho, 2020; Khattab and Zaharia, 2020; Zhan et al., 2020b,a). Referred to as pair-wise training, the model is asked to score the similarity between the query embedding and a ground truth relevant (positive) document embedding, higher than the one between the query and a negative document embedding.

While such contrastive learning approaches are effective, by only considering pairs of positive and negative documents at a time, they (1) deviate from the target objective of comparing a query against many documents while discarding inter-document information, and (2) depart from core list-wise evaluation metrics like nDCG (Calauzènes et al., 2012).

Addressing the first shortcoming, recent works have employed "in-batch" negatives: given a batch containing *(query, positive document)* tuples, the negatives to a tuple are set as the known positive documents from other queries within that batch. (e.g. Karpukhin et al. (2020); Luan et al. (2021); Zhan et al. (2020b); Hofstätter et al. (2021); Qu et al. (2021)). Although this approach efficently increases the number of negatives to improve performance, the presence of *hard* negative samples[1] is critical in achieving state-of-the-art (SOTA) (Xiong et al., 2020; Zhan et al., 2021b; Qu et al., 2021; Hofstätter et al., 2021).

The rich literature on *learning-to-rank (L2R)* has outlined compelling reasons for taking into account the context of other candidate documents being ranked for a query when scoring each document (Cao et al., 2007; Ai et al., 2019, 2018), realized in various *list-wise* optimizations. Such approaches allow for the model to directly optimize IR metrics as opposed to a surrogate pair-wise loss as seen in the contrastive training regime. Despite these list-wise approaches achieving competitive results in a variety of ranking situations, considerations of computational complexity and stochastic stability practically relegate them to shallow neural models over handcrafted feature vectors (Bruch et al., 2020; Pang et al., 2020; Chen and Eickhoff, 2021).

[1]Hard negatives look similar in topic and term distribution to relevant documents, while not actually satisfying the information need.

In this paper, we extend existing work in negative sampling and list-wise learning to allow for large pre-trained language models to take advantage of the context found in a large, *coherent* set of candidate documents. We particularly examine the effect of constituent parts of the query context, i. e. (1) jointly scoring a large number of negatives, (2) using retrieved (query-specific) instead of random negatives, and (3) a fully list-wise loss. To this end, we introduce *COntextual Document Embedding Reranking (CODER)*, a highly efficient and generic fine-tuning framework that for the first time enables incorporating context, previously only considered in learning-to-rank neural networks, into transformer-based language models used in state-of-the-art dense retrieval. CODER acts as a lightweight performance enhancing framework that operates on precomputed document embeddings, while transforming the query to account for new list-wise context information over a large number of query-specific hard negative candidate documents. It can be applied to virtually any existing dual-encoder model, and used both for single- as well as two-stage dense retrieval.

Our contribution is three-fold: (1) We introduce an efficient framework which enables leveraging ranking context; (2) We conduct a large set of experiments on the MS MARCO (Bajaj et al., 2018) and TripClick (Rekabsaz et al., 2021b) collections and show that CODER can considerably enhance the effectiveness of a wide class of dense retrieval models at minimal computational cost, while achieving new SOTA results on TripClick; (3) We explore the impact of the constituent parts of ranking context in learning effective models. Our code and trained resources are available in `https://github.com/gzerveas/CODER`.

2 Related Work

Recent work has demonstrated the importance of the quality of negative documents used during fine-tuning. Xiong et al. (2020) periodically re-encode every query and document in the collection during training in order to mine the most difficult documents to use as negative candidates via approximate nearest neighbor (ANN) search. Improving on this slow and resource-intense process, Zhan et al. (2020a) (published as Zhan et al. (2021b)) forego fine-tuning of the document encoder, instead only fine-tuning the query encoder while dynamically mining negatives. TAS-B (Hofstätter et al., 2021) also improves the quality of negatives by clustering semantically similar queries, such that the in-batch negatives are indirectly related to the ground truth document. While we contribute to this line of research, we show that one can avoid such complexity and directly benefit from list-wise optimization applied on a large, coherent and informative context of candidate documents, retrieved for each query in advance.

Moving from quality to quantity of negative candidates, RocketQA (Qu et al., 2021) drastically increases the quantity of random in-batch negatives to several thousands by sharing negatives across at least 8 V100 GPU instances. Despite its huge size, this pool of sampled negatives includes only 4 retrieved hard negatives per query and is otherwise almost entirely random, with no shared context across documents. Unlike CODER, this approach necessitates training an expensive cross-encoder model to "denoise" (filter out) retrieved candidates, otherwise yielding poor performance. Moreover, it involves using the cross-encoder to pseudo-label additional data samples, an approach adopted by the currently top performing dual encoder models following up this work, RocketQAv2 (Ren et al., 2021b), which additionally leverages list-wise cross-encoder teaching, and PAIR (Ren et al., 2021a), which includes a loss term capturing similarity between passages.

To address the increasing complexity and computational requirements of training pipelines, Gao and Callan (2021a,b) instead propose corpus-specific, self-supervised pre-training with a bespoke transformer add-on (see Baselines in Section 4). An orthogonal approach to reduce computational requirements focuses on jointly optimizing query optimizer and product quantization for ANN search (Zhan et al., 2021a).

List-wise loss functions have been extensively used within learning-to-rank (L2R), although they have been limited to either shallow neural models (Bruch et al., 2020; Cao et al., 2007) or various deep networks Pasumarthi et al. (2019a); Ai et al. (2018, 2019); Pang et al. (2020) in works focusing on L2R datasets. These consist of handcrafted feature vectors representing query-document similarity such as term overlap, click-through rate, BM25 scores, and other salient features. Effectively applying L2R concepts to transformer-based language models used for ad-hoc dense retrieval is a non-trivial challenge, and represents CODER's exten-

sion of prior works.

Finally, there is a body of work utilizing large pre-trained language models for retrieval in cross-encoder (Nogueira and Cho, 2020; Lin et al., 2021; Hofstätter et al., 2022), late cross-encoder (Khattab and Zaharia, 2020; Santhanam et al., 2021), and generative rankers (Lesota et al., 2021), as well as query/document expansion and indexing (Zheng et al., 2020; Naseri et al., 2021; Mallia et al., 2021; Nogueira et al., 2020; Gao et al., 2021). Co-BERT (Chen et al., 2022), a recent method leveraging ranking context, uses a cross-encoder BERT Reranker to select candidates and to compute feature vectors as input for the L2R methods above (Pang et al., 2020) through query-document term interactions. In comparison, CODER is orders of magnitude more efficient both during training and inference.

All existing approaches either advocate for using a handful of hard negatives (e.g. (Karpukhin et al., 2020)), or compromise with it due to computational constraints. By recognizing the importance of context, our proposed framework is the first to allow the combination of quantity and quality of negatives for training SOTA dual encoder models with very modest computational resources.

3 Method

CODER involves fine-tuning a pre-trained query encoder to learn a query representation that is as proximal as possible to the representation of the ground-truth relevant document(s), by adjusting it to better account for the context of multiple query-related documents. The architecture consists of two main components (Fig. 1): a query encoder, which builds a query representation, and the document set scoring module, which, given a query representation, jointly scores a set of N precomputed embeddings of positives and hard negatives retrieved by an arbitrary retrieval method, M_C. Using precomputed document embeddings reduces computational costs (memory, FLOPs) by a factor of N. Thus, unlike all existing approaches, we can afford to use a large number of such hard negatives ($N = 1000$ in our experiments, unless otherwise noted).

3.1 Architecture

3.1.1 Query Encoder

The query encoder can be any pre-trained transformer encoder, such as BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2020), RoBERTa (Liu et al., 2019), or ERNIE (Zhang et al., 2019). We initialize its weights from an existing model already fine-tuned for retrieval, which we call "*base model*", M_D.

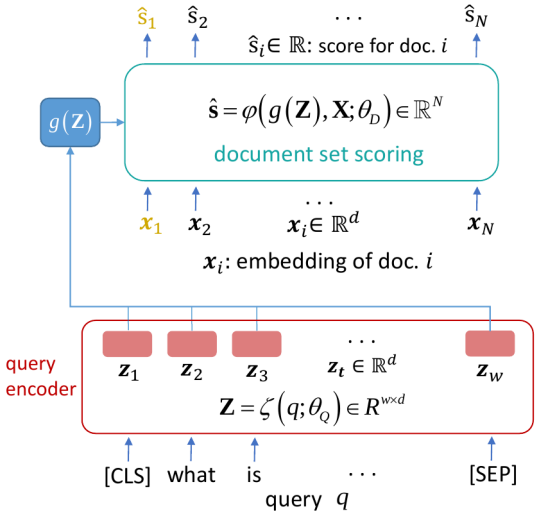

Figure 1: Schematic diagram of the CODER method. The architecture consists of two main components: the query encoding module, which embeds a tokenized query, and the document set scoring module, which jointly scores a set of N precomputed candidate document embeddings. Here, we experiment specifically with $\varphi = \mathbf{X} \cdot g(\mathbf{Z})$, combined with a list-wise loss and large $N = 1000$, which we show allows to effectively leverage ranking context.

Formally, for each query token q_t, $t \in \mathbb{N} : 1 \leq t \leq w$, where w is the length of the tokenized query sequence, it extracts a vector representation $\mathbf{z}_t \in \mathbb{R}^d$, where d is the encoder's internal representation dimension. These vectors can be linearly projected to a space of different dimensionality and become $\mathbf{z}'_t \in \mathbb{R}^{d'}$, to match the dimensionality of the document embeddings d', in case the latter differs. In the general case we thus denote the extracted representation of a query q as:

$$\mathbf{Z}' = [\mathbf{z}'_1; \ldots; \mathbf{z}'_w] = \zeta(q; \theta_Q) \in \mathbb{R}^{w \times d'}, \quad (1)$$

where θ_Q are the parameters of the query encoder. The individual token representations are aggregated into a single vector using an aggregation function g. In our experiments, we let $g(\mathbf{Z}') = \frac{1}{w} \sum_t \mathbf{z}'_t$ be the mean when using RepBERT (Zhan et al., 2020b), and $g(\mathbf{Z}') = \mathbf{z}'_1$ (i.e. the representation of the `[CLS]` token) when using TAS-B (Hofstätter et al., 2021) as the base model for implementing the query encoder ζ (see Section 4).

3.1.2 Document scoring function

The document set scoring function is represented by φ which produces a set of N scalar relevance scores $\hat{s}_i \in \mathbb{R}$, $i \in \mathbb{N} : 1 \leq i \leq N$. It takes as an input the aggregated query representation $g(\mathbf{Z}')$ from the previous section, and a set of N document embeddings $\mathbf{x}_i \in \mathbb{R}^m$, $i \in \mathbb{N} : 1 \leq i \leq N$, precomputed by the base model. Using learnable linear projections, the dimensionality of the document embeddings can potentially be changed to accommodate different scoring functions (e.g., transformer blocks with an internal representation dimension $d' \neq m$), while matching the dimensionality of the query embeddings. Succinctly, the output relevance scores are:

$$\hat{\mathbf{s}} = \varphi\left(g(\mathbf{Z}'), \mathbf{X}; \theta_D\right) = \mathbf{X} \cdot g\left(\mathbf{Z}\right) \in \mathbb{R}^N, \quad (2)$$

where $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_N] \in \mathbb{R}^{N \times m}$ are the N document embeddings, θ_D are the parameters of the scoring function, and $d' \equiv d$ (i.e. query and document embeddings have the same dimensionality).

While a variety of functions can be used as a scoring function in our framework, including transformers (see Appendix Section A.4), for all results presented in this work we leverage the simple inner product, which interestingly achieves significant performance improvements even without the contextualized transformation of document embeddings. It thus appears that jointly scoring a large number of query-specific candidates for the same query within a list-wise loss establishes a strong enough context for improving performance (see Section A.5). Beyond computational efficiency, the main advantage of the above function is that it facilitates directly using the fine-tuned query encoder for dense retrieval (single-stage) through fast approximate nearest neighbor search. Instead, a non-linear scoring module would only allow using the model for reranking in a two-stage retrieval setting (candidate retrieval, followed by reranking).

3.2 Training through CODER

The document representations $\mathbf{X}$ are precomputed using the *document* encoder part of any state-of-the-art dual encoder retrieval model M_D. To accelerate training convergence, we initialize our query encoder ζ from the query encoder of the same dual encoder retrieval model. Throughout training, the parameters θ_Q of the query encoder (and θ_D of the scoring function φ, if the latter is learnable) are fine-tuned. To support a memory and compute-efficient training setting without the need for large or parallel GPUs, the document representations $\mathbf{X}$ remain fixed, although in general they can be transformed by function φ before computing the document similarity scores. As with all dense retrieval methods (e.g., (Zhan et al., 2020b; Xiong et al., 2020; Zhan et al., 2021b; Hofstätter et al., 2021)), document representations are assumed to have been precomputed and indexed for fast inference runtimes.

A key difference between CODER and all dense retrieval methods is that, for each query, along with the k positive (ground-truth) documents, the model is trained to jointly score a *large number* $N - k$ of top candidate documents retrieved by some *candidate* retrieval method M_C. We note that M_C does not need to be the same method as the base method that provides the document representations and the query encoder. This potentially allows leveraging methods with different characteristics (e. g. methods with higher recall versus precision, or a lexical overlap / sparse representation such as BM25), and can prove beneficial, as shown in Section 5.

3.3 Loss function

To best take advantage of jointly scoring N documents for each query, we choose the ListNet loss (Cao et al., 2007), which is the KL-divergence between a distribution over the predicted scores $\hat{\mathbf{s}}$ (given by Eq. (2)) for the N candidate documents, and a distribution over the target (ground-truth) relevance labels $\mathbf{y} \in \mathbb{R}^N$, given by the dataset for the same set of candidates (the relevance score of positive documents is a positive scalar, while for negative or documents whose label is not explicitly defined it is set to $-\infty$):

$$\begin{aligned} \mathcal{L}\left(\mathbf{y}, \hat{\mathbf{s}}\right) &= \mathrm{D}_{\mathrm{KL}}\left(\sigma(\mathbf{y}) \;||\; \sigma(\hat{\mathbf{s}})\right) \\ &= -\sum_{i=1}^{N} \sigma(\mathbf{y})_i \log \frac{\sigma(\hat{\mathbf{s}})_i}{\sigma(\mathbf{y})_i} \end{aligned} \quad (3)$$

where σ denotes the softmax function. Jointly scoring a large number of *retrieved* candidates for each query, in combination with the KL-divergence loss, distinguishes our method from existing dense retrieval methods. This combination establishes and exploits a *context* for each query and it is key for obtaining a performance improvement over the base method, as we show in Section 5.4 (and further discuss in Appendix Section A.5). The benefit is expected to be even greater for datasets which include multiple document labels per query, option-

ally defined over several levels of relevance. We show such results in Section 5.2.

4 Experimental Setup

Datasets. We conduct the experiments on passage and document retrieval tasks,[2] using two large publicly available IR collections in the domains of web and health retrieval. The first dataset is the MS MARCO Passage Retrieval dataset (Bajaj et al., 2018), used for training and evaluation. We evaluate the models trained on MS MARCO also on TREC Deep Learning Passage Retrieval track 2019 and 2020 (Craswell et al., 2020, 2021). The second dataset used for training and evaluation is TripClick, a recently introduced health document retrieval dataset (Rekabsaz et al., 2021b). Details of the collections are provided in Section A.2 of the Appendix. While the training data of MS MARCO contains only approximately 1 relevance judgement per query, the TripClick collection has the advantage of providing a much larger set of relevance information, namely approximately 42, 9, and 3 data points per query in HEAD, TORSO, and TAIL sets, respectively. As shown in the next section, this is particularly beneficial when optimizing over a large ranking context in a list-wise manner. Evaluation details are given in Section A.3 of the Appendix.

Baselines. We choose several dense retrieval models as baselines, i. e. "base models" subjected to CODER fine-tuning:
1. RepBERT (Zhan et al., 2020b), a BERT-based model with a typical dual encoder architecture which underpins all state-of-the-art dense retrieval methods, trained using a triplet Max-Margin loss.
2. TAS-B (Hofstätter et al., 2021), which, besides being a top-performing dense retrieval method on the MS MARCO / TREC-DL 2019, 2020 datasets, it also represents methods that have been optimized with respect to their training process (details in Section 2).
3. Finally, to explore the limits of CODER, we use it to fine-tune a trained CoCondenser retriever (Gao and Callan, 2021b), the state-of-the-art dense retrieval model *that does not* make use of query-document term interactions, cross-encoder teacher models or additional pseudo-labeled data samples, but instead relies on extensive corpus-specific, self-supervised pre-training using a special architecture and contrastive loss component. It is a particularly challenging baseline for our CODER framework, because it has been trained through (a) mining for hard negatives using a trained version of the model itself, and (b) the Negative LogLikelihood (InfoNCE) loss, which is "nearly" list-wise (it differs from our KL-divergence loss only when there are more than one positive candidates).

Configurations. For the CODER framework, we use the following notation: $M_F \rightarrow \text{CODER}(M_D, M_C)$, where M_D is the base model used to encode documents into document embedding vectors (and initialize the query encoder weights), M_C is the first-stage retrieval method used to procure the candidate (context) documents reranked during the CODER training process, and M_F is the retrieval method used as a first stage when CODER is used as a reranking method during inference; M_F vanishes in case CODER is used directly for single-stage dense retrieval, and the notation $\text{CODER}(M_D, M_C)$ is used instead. In addition to TAS-B and RepBERT, we also experiment with BM25 (Anserini implementation (Yang et al., 2017)) as M_C and M_F methods. Specific hyperparameter details can be found in Appendix Table 3.

5 Results

5.1 Results on MS MARCO and TREC DL

After only a fast and efficient fine-tuning (3.5 hours for TAS-B, 4.5 hours for RepBERT on a single NVIDIA TITAN RTX GPU), we observe a substantial performance benefit when applying our CODER framework to TAS-B and RepBERT, as seen in Table 1.

CODER improves retrieval performance remarkably compared to both the original (single-stage) RepBERT, as well as two-stage cascade BM25→RepBERT. CODER confers the largest performance benefit on RepBERT when reranking BM25 candidates in a cascade. The single-stage retriever fine-tuned through CODER is much improved compared to the original RepBERT, and almost as effective as the cascade, without introducing any latency or complexity.

Our framework also significantly improves the performance of the highly optimized TAS-B method, which leverages hard negatives and dual knowledge distillation from two powerful cross-encoder models, BERT-Reranker and ColBERT. We furthermore observe that using the CODER-

[2] When referring to the unit of retrieval we use the terms "passage" and "document" interchangeably.

Model	MS MARCO dev		TREC DL 2019		TREC DL 2020		Latency
	MRR@10	nDCG@10	MRR@10	nDCG@10	MRR@10	nDCG@10	(ms/query)
BM25 [Anserini]	0.187	0.234	0.843 / 0.682	0.497 / 0.417	0.820 / 0.655	0.488 / 0.412	50[1]
L2Re(ANCE) (Zhan et al., 2020a)	0.341	-	-	0.675	-	-	47
Co-BERT* (Chen et al., 2022)	-	-	0.958	0.700	0.839	0.699	> 1000
[2]ColBERT*v1; v2	0.360; 0.397	-	-	-	-	-	458
[2]BM25 → ColBERT*	0.349	-	-	-	-	-	[BM25] + 61
[3]RocketQAv1; v2	0.370; 0.388	-	-	-	-	-	-
CoCondenser [our evaluation]	0.381	0.446	0.971 / 0.879	0.715 / 0.656	0.937 / 0.833	0.680 / 0.618	≃ [RepBERT]
RepBERT (abbrev: RB) [our eval.]	0.304	0.359	0.917 / 0.766	0.616 / 0.548	0.902 / 0.763	0.621 / 0.561	70
BM25 → RepBERT	0.317	0.373	0.969 / 0.795	0.674 / 0.593	0.893 / 0.781	0.640 / 0.579	[BM25] + 5.8
BM25 → CODER(RB, BM25)	0.326	0.384	0.953 / 0.798	0.675 / 0.600	0.914 / **0.816**	0.654 / 0.593	[BM25] + 5.8
BM25 → CODER(RB, RB)	**0.327**^{ab}	**0.385**^{ab}	0.953 / **0.806**	0.677 / **0.603**^{a}	0.898 / 0.787	0.672 / **0.611**^{ab}	[BM25] + 5.8
RB → CODER(RB, RB)	0.324	0.383	0.905 / 0.785	0.650 / 0.593	0.918 / 0.785	0.660 / 0.598	[RepBERT] + 5.8
CODER(RB, BM25)	0.311	0.368	0.855 / 0.750	0.606 / 0.552	0.906 / 0.790	0.603 / 0.550	[RepBERT]
CODER(RB, RB)	0.325	0.384	0.905 / 0.785	0.652 / 0.593	0.918 / 0.785	0.660 / 0.598	[RepBERT]
TAS-B (Hofstätter et al., 2021)	0.340	0.402	0.892	0.712	0.843	0.693	64
TAS-B [our evaluation]	0.344	0.408	0.951 / 0.875	0.721 / 0.659	0.921 / 0.832	0.685 / 0.620	< 50
BM25 → TAS-B	0.343	0.404	0.971 / 0.857	0.723 / 0.648	0.918 / 0.838	0.696 / **0.633**	[BM25] + 5.5
BM25 → CODER(TAS-B, BM25)	0.349	0.409	0.983 / 0.872	0.727 / 0.654	0.935 / **0.846**	0.690 / 0.629	[BM25] + 5.5
BM25 → CODER(TAS-B, TAS-B)	0.350	0.411	0.971 / 0.828	0.728 / 0.654	0.926 / **0.846**	0.693 / **0.630**	[BM25] + 5.5
TAS-B → CODER(TAS-B, TAS-B)	**0.355**^{ab}	**0.419**^{ab}	0.966 / 0.857	0.728 / **0.668**	0.923 / **0.844**	0.686 / 0.623	[TAS-B] + 5.5
CODER(TAS-B, BM25)	0.347	0.409	0.965 / **0.890**	0.723 / 0.665	0.934 / 0.835	0.678 / 0.612	[TAS-B]
CODER(TAS-B, TAS-B)	**0.355**^{ab}	**0.419**^{ab}	0.966 / 0.857	0.728 / **0.668**	0.923 / **0.844**	0.686 / 0.623	[TAS-B]

Table 1: Performance for passage ranking when applying CODER to the RepBERT (middle section) and TAS-B (bottom section) base methods. In the notation $M_F \rightarrow \text{METHOD}(M_D, M_C)$: M_F is the method used for first stage retrieval when using METHOD for reranking, M_D is the base method and M_C is the retrieval method which provides the context (candidate) passages during training. **Bold font denotes best results within the same section (separated by continuous rules)**. Results of the statistical significance tests (paired t-test) are reported only for the best performing models, where the symbols ^{a} and ^{b} denote a significant improvement ($p < 0.05$) with respect to the base M_D and BM25$\rightarrow M_D$, respectively. For TREC DL, two values are given for each metric separated by a slash, corresponding to the lenient / strict (official) interpretation of relevance labels. Models with * use cross-encoder term interactions. Rows with [2] are from Khattab and Zaharia (2020); Santhanam et al. (2021), and with [3] from (Qu et al., 2021; Ren et al., 2021b).

trained query encoder directly for single-stage dense retrieval is exactly as effective as using CODER to rerank TAS-B candidates in a cascade, on both MS MARCO and TREC DL tracks. This result suggests that under certain conditions, CODER can generalize its ranking function from the provided training context (limited set of fixed hard negatives) to the entire dataset, even without the use of dynamic negative mining, huge batch sizes or "denoising" as seen in previous work (Qu et al., 2021; Xiong et al., 2020).

Putting these results into context, we can see that simply by training through contextual reranking, a dual encoder model such as TAS-B can improve to the point that its reranking effectiveness is higher than a powerful model such as ColBERT, a term-interaction model still fast enough to be practically considered for real-time reranking, but at a fraction of the reranking latency cost (5.5 ms vs 61 ms, i.e., less than 1/10th). Its single-stage ranking performance approaches the one by ColBERT, while being about 10x faster (about 50 ms vs. 458 ms), making it a top-performing method within its latency class.

Finally, to test the limits of CODER (see Section 4), we apply it to fine-tuning a trained CoCondenser retriever, the SOTA dense retrieval model that does not rely on using a cross-encoder in its pipeline. We observe a slight improvement of 0.002 MRR@10 and 0.004 Recall@10 (the latter statistically significant) on the MS MARCO validation set. This smaller improvement on MS MARCO is expected, given that it is a dataset providing very few relevance judgements per query and thus (a) poor context for training, and (b) an evaluation setting that may be unsuitable to resolve differences in ranking effectiveness for a contextually-trained model. For this reason, we additionally evaluate the models on TripClick.

5.2 Results on TripClick

Following Rekabsaz et al. (2021b), we report performance in terms of MRR@10, nDCG@10 and Recall@10; nDCG@10 is considered the most important metric, as multiple relevant documents per query exist, and the DCTR relevance set additionally uses multiple levels of relevance. The results are presented in Table 2: Using the same hyperparameters as in MS MARCO, CODER fine-tuning

Model	MRR@10	nDCG@10
BM25[1]	0.276	0.224
Transformer-Kernel[1]	0.434	0.284
BERT-Dot (SciBERT)[2]	0.530	0.243
BERT-Cat (SciBERT; PMBERT)[2]	0.595; 0.582	0.294; 0.298
RepBERT (abbrev: RB)	0.526	0.255
BM25 → RepBERT	0.538	0.262
RB → CODER(RB, RB)	**0.637***	**0.318***
CODER(RB, RB)	0.634	0.316

Table 2: Performance when applying CODER to RepBERT on the TripClick HEAD dataset, using multi-level (DCTR) relevance labels (metrics cut-off of 10). All CODER results are statistically significant (paired t-test, $p < 0.05$) with respect to both the base and BM25→base methods. The symbol * on best results denotes statistically significant improvement with respect to all baselines. Results with [1] are from Rekabsaz et al. (2021b), with [2] from Hofstätter et al. (2022).

tremendously improves the performance of RepBERT trained on TripClick, both in reranking as well as in single-stage dense retrieval. The improvement is especially pronounced on the HEAD subset, where many relevance judgements per query are available. CODER achieves the SOTA performance by a large margin, ahead of all published results in the literature and on the TripClick leaderboard[3], including ensembles of heavyweight cross-encoder BERT Rerankers which were pre-trained on the domain-specific PubMedBERT (medicine) and SciBERT (science) corpora (Hofstätter et al., 2022). CODER also significantly outperforms the best existing dense retrieval method (BERT-Dot pre-trained on SciBERT (Hofstätter et al., 2022)) on the TORSO and TAIL subsets (see Appendix Table 4). However, presumably because these subsets consist of rare queries with significantly fewer relevance judgements per query, it falls behind the domain-pretrained cross-encoders.

Finally, we use TripClick's validation and test sets purely for "zero-shot" evaluating the RepBERT, CODER(RepBERT), CoCondenser and CODER(CoCondenser) models trained exclusively on MS MARCO. We emphasize that, uniquely in this setting, these models have not been trained or fine-tuned on TripClick; they are the same models described in Section 5.1, now evaluated on a large dataset with severe distribution shift (biomedical domain). While zero-shot evaluation is used increasingly often to demonstrate the effectiveness and generalizability of large transformer models (e.g. (Brown et al., 2020; Hao et al., 2022; Radford et al., 2021)), here we use it as a means to bypass the challenge of the expensive pre-training and fine-tuning of CoCondenser on TripClick. Results are shown in Appendix Table 5: naturally, we observe that zero-shot performance in absolute terms is low; however, CODER-trained models perform always better. Moreover, the performance order CODER(CoCondenser) > CoCondenser > CODER(RepBERT) > RepBERT, consistent with our observations for MS MARCO, holds also here in almost every comparison.

[3]https://tripdatabase.github.io/tripclick/

5.3 Efficiency

What is the additional cost of using CODER? Using a single NVIDIA TITAN RTX GPU (on a node with Intel Xeon Gold 6142 CPU), it takes about 186 ms to rank 1000 candidates per query in a batch of 32 queries (out of which less than 10ms refer to computing representations and scores, with the rest taken up by batching and loading samples to the GPU), i.e., a latency of 5.5-5.8 ms per query is introduced when using CODER as a second-stage reranker in a cascade. Using CODER as a single-stage dense retriever only requires the same processing time as the base method, e.g. RepBERT or TAS-B. Table 1 reports the latency reported in (Zhan et al., 2020b), but this in practice will be determined by the time for loading the query sequence to the GPU and encoding it (in our setup, approx. 5.4 ms per query, out of which approx. 0.3 ms is the time to compute the representation), in addition to the time for finding the approximate nearest neighbors using a library such as FAISS (Johnson et al., 2017).

5.4 Analysis of key factors

5.4.1 Importance of context

In this section we wish to assess the intuitions that scoring a document within a context of other documents related to the same query can be advantageous, and that a list-wise loss function like the one we present in Section 3.3 is better equipped to leverage this context compared to a superposition of separate pair-wise loss components. We first study the impact of varying the type and number of negative documents during training.

In Figure 2 we show how the performance of a model initialized from a trained RepBERT base model evolves during fine-tuning through CODER, measured in MRR@10 on our MS MARCO validation set when reranking 1000 candidates first retrieved for each query by BM25. Different

curves correspond to different training settings. The leftmost evaluation point corresponds to the best model checkpoint achievable through standard triplet-based training.

Composition of negatives: We observe that training with only randomly sampled negatives, even in large numbers, leads to a deterioration of performance. We note that this is the case because through CODER we are optimizing a baseline model which has already been trained to peak performance through random negative documents. Using a small number of retrieved candidates together with in-batch random negatives, as in current SOTA methods (Zhan et al., 2020a; Qu et al., 2021), again leads to deteriorating performance, even with many random negatives. Only by using a large number of coherent, query-specific negative samples (i.e. retrieved by a retrieval method) for the same query can CODER extract additional performance from the baseline. Moreover, adding 1000 additional random documents per query (in-batch negatives) on top of those does not yield any benefit, presumably because they are not adding any context (as they are unrelated to the query and documents under assessment) and/or they are not sufficiently challenging.

List-wise loss: When training with the most commonly used pair-wise loss (Multi-Label Max Margin, purple dashed curve), even when using 1000 retrieved candidates as negatives within exactly the same training setup, i.e. including the same positive and negative examples for the same query in the same batch, we observe that performance only modestly improves performance (+0.009 MRR@10). The improvement can be attributed to the fact that the model now encounters a large number of coherent, query-specific documents for the same query during a single step of training, which offers a more complete and accurate view of the loss landscape and thus leads to a more accurate update of parameters. However, the ranking context is exploited more effectively when using a list-wise KL-divergence loss (dark blue curve), as CODER fine-tuning improves MRR@10 from 0.345 to 0.363 (+0.018).

Finally, we note that the training loss continuously decreased throughout training in all settings mentioned above (see Figure 5 in the Appendix), including the ones in which performance on the validation set was deteriorating at the same time (Figure 2). This fact suggests that deteriorating

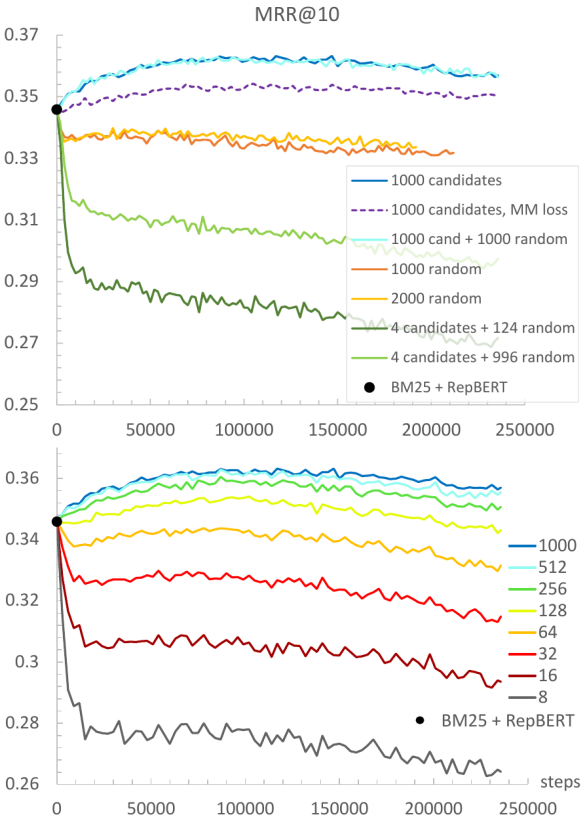

Figure 2: Performance of BM25→CODER(RepBERT) on MS MARCO validation set over training steps. The left-most point corresponds to reranking BM25 candidates using the fully trained RepBERT. **Top**: Effect of type and number of documents used as negatives. The purple dashed curve corresponds to training with a pair-wise (Max Margin) loss. **Bottom**: Effect of number of BM25 candidates used as negatives during training.

performance in those settings can be interpreted as overfitting, when there is insufficient information/signal captured by the training objective in order to learn to rank more effectively.

5.4.2 Number of context documents

We now investigate the importance of the quantity of negative documents per query during training. In Figure 2, where the number of BM25-retrieved candidate negatives is adjusted, we observe increasing performance per additional document until we reach a number in the order of the dimensionality of the embedding space (here, 768 for BERT-base). Increasing the number of negatives beyond this point yields diminishing returns (see section A.1 for an explanation).

We note that these numbers of negative *candidates* (as opposed to random documents) per query are much higher than the ones used in all contemporary work (max. 4 candidates have been employed in Qu et al. (2021) and 30 in Gao and

Callan (2021b)); the reason that such a high number is necessary in order to achieve a performance improvement is that we are fine-tuning a model already trained to saturation. Only a large number of retrieved candidate negatives provides enough training signal to overcome overfitting and improve performance (Figure 6 in the Appendix shows the training loss to decrease in all cases).

The above results suggest that most dense retrieval models would likely benefit from training using a context, i.e., a large number of retrieved candidate documents per query, combined with an appropriate list-wise loss.

6 Conclusion

We examine the importance of ranking context and the effect of its constituent parts, i.e., a fully list-wise loss, a large number of negatives, and retrieved (query-specific) instead of random negatives, and show that they are all important ingredients for improving performance. We demonstrate that a lightweight reranking framework designed to leverage context is sufficient to significantly enhance the effectiveness of a wide range of dense retrieval models, without expensive cross-encoder distillation, pseudo-labeling or "denoising" negatives. The only computational overhead is a fast, resource-light fine-tuning process, with little (when the model is used as a reranker) to no (when used as a single-stage retriever) extra computational cost during inference.

7 Limitations

In the present work we have endeavored to demonstrate that the performance benefits achieved by our framework are generalizable with respect to the dense retrieval method used as base, as well as the document collections to be searched. For this purpose, we selected as baselines dual encoder models which are highly representative of the state-of-the-art IR approaches, including ones that have undergone highly optimized training processes.

However, it is always possible that some retrieval models will not enjoy similar performance enhancement when trained through CODER, owing to the large variety of modeling assumptions or specially required conditions for training different models. For example, a model trained through a process already leveraging most of the ranking context elements examined here (list-wise loss, query-specific/retrieved candidates, a large number of them) will stand to benefit less. Conversely, models trained on datasets including multiple positive documents per query will exhibit better performance.

We further emphasize that the retrieval methods compatible with our approach make use of *distinct* document and query representations. Thus, it is not possible to use CODER to directly fine-tune term-interaction (a.k.a. cross-encoder) models like BERT Reranker, because each document within CODER is only available as a single embedding vector. Finally, we note that, although it is theoretically possible to initialize the query encoder from a generic NLP language model, in all our presented experiments it is initialized from a base model (query/document encoder) already trained for retrieval, the one that is also used to precompute document embeddings.

Acknowledgements

G. Zerveas would like to thank the Onassis Foundation for supporting this research. This work received financial support from the State of Upper Austria and the Federal Ministry of Education, Science, and Research, through grant LIT-2021-YOU-215, and is also supported in part by the NSF (IIS-1956221). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF or the U.S. Government.

References

Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 135–144, New York, NY, USA. Association for Computing Machinery.

Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. *arXiv:1811.04415 [cs]*. ArXiv: 1811.04415.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]*. ArXiv: 1611.09268.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind

Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. ArXiv:2005.14165 [cs].

Sebastian Bruch, Shuguang Han, Mike Bendersky, and Marc Najork. 2020. A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM 2020)*, pages 61–69.

Christopher J. C. Burges. 2010. From ranknet to lambdarank to lambdamart: An overview.

Clément Calauzènes, Nicolas Usunier, and Patrick Gallinari. 2012. On the (non-)existence of convex, calibrated surrogate losses for ranking. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA. Association for Computing Machinery.

Xiaoyang Chen, Kai Hui, Ben He, Xianpei Han, Le Sun, and Zheng Ye. 2022. Incorporating Ranking Context for End-to-End BERT Re-ranking. In *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 111–127, Cham. Springer International Publishing.

Zhizhong Chen and Carsten Eickhoff. 2021. Poolrank: Max/min pooling-based ranking loss for listwise learning & ranking balance. *ArXiv*, abs/2108.03586.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. *arXiv preprint arXiv:2102.07662*.

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

Luyu Gao and Jamie Callan. 2021a. Condenser: a pre-training architecture for dense retrieval. In *EMNLP*.

Luyu Gao and Jamie Callan. 2021b. Unsupervised corpus aware language model pre-training for dense passage retrieval. *ArXiv*, abs/2108.05540.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. 2022. Language Models are General-Purpose Interfaces. ArXiv:2206.06336 [cs].

Sebastian Hofstätter, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Establishing Strong Baselines for TripClick Health Retrieval. *arXiv:2201.00365 [cs]*. ArXiv: 2201.00365.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and A. Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *SIGIR*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. *undefined*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv:1702.08734 [cs]*. ArXiv: 1702.08734.

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*, pages 39–48.

Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. *A Modern Perspective on Query Likelihood with Deep Generative Retrieval Models*, page 185–195. Association for Computing Machinery, New York, NY, USA.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4):1–325.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *arXiv:2010.11386 [cs]*. ArXiv: 2010.11386.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019.

RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. ArXiv: 1907.11692.

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.

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

Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval t. *Foundations and Trends® in Information Retrieval*, 13(1):1–126.

Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. 2021. Ceqe: Contextualized embeddings for query expansion. In *European Conference on Information Retrieval*, pages 467–482. Springer.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage Re-ranking with BERT. *arXiv:1901.04085 [cs]*. ArXiv: 1901.04085.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718.

Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 499–508, New York, NY, USA. Association for Computing Machinery.

Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019a. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2970–2978, New York, NY, USA. Association for Computing Machinery.

Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019b. Self-Attentive Document Interaction Networks for Permutation Equivariant Ranking. *arXiv:1910.09676 [cs]*. ArXiv: 1910.09676.

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. *arXiv:2010.08191 [cs]*. ArXiv: 2010.08191.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. ArXiv:2103.00020 [cs].

Navid Rekabsaz, Simone Kopeinik, and Markus Schedl. 2021a. Societal biases in retrieved contents: Measurement framework and adversarial mitigation for bert rankers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. 2021b. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2513. Association for Computing Machinery, New York, NY, USA.

Navid Rekabsaz and Markus Schedl. 2020. Do neural ranking models intensify gender bias? In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2065–2068.

Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021a. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183. ArXiv: 2108.06027.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021b. 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.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*. ArXiv: 1910.01108.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. ArXiv:2112.01488 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*. ArXiv: 2007.00808.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1253–1256, New York, NY, USA. Association for Computing Machinery.

Dianhai Yu Yanjun Ma and Dianhai Yu Yanjun Ma. 2019. PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. *Frontiers of Data and Domputing*, 1(1):105–115.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021a. *Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance*, page 2487–2496. Association for Computing Machinery, New York, NY, USA.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021b. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada. ACM.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020a. Learning To Retrieve: How to Train a Dense Retrieval Model Effectively and Efficiently. *arXiv:2010.10469 [cs]*. ArXiv: 2010.10469.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020b. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv:2006.15498 [cs]*. ArXiv: 2006.15498.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. 2020. BERT-QE: Contextualized query expansion for document re-ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4718–4728.

A Appendix

A.1 Motivation for simultaneously scoring a large number of negative documents

In contrastive learning, models are trained to assign a higher score for similarity between the query and a positive document than between the query and all negative documents: $s(q, p_j^+) > s(q, p_i^-),\ \forall i, j$, where p_j^+ denotes a positive and p_i^- a negative document/passage respectively. In Figure 3, vector representations of documents are depicted as points (red for positive, blue for negative documents) in a d-dimensional space that is shared with the query representation vector. Because functions used to compute similarity increase with decreasing Euclidean distance, the objective can be fulfilled by learning to map the query within a smaller distance from a given positive document p_j^+ compared to all negative documents[4]. However, for a space of dimension d, when fewer than $d+1$ negative documents are included in a loss calculation, there is an infinite subspace where the query representation can lie, arbitrarily far from the positive document, and still satisfy this condition. At least $d+1$ negative documents are required to constrain the space of objective-favored query mappings to a bounded convex polytope: given that the positive document is contained within a simplex formed by $d+1$ negative documents as vertices in that space (interval in $\mathbb{R}$, triangle in $\mathbb{R}^2$, tetrahedron in $\mathbb{R}^3$ etc), the loss function will favor mapping the query onto another simplex in the same space, within which the distance from the positive document will be bounded. Of course, training with fewer than $d+1$ negative documents per query still works: although the loss landscape as revealed by only a few negative documents is a rough approximation, and thus the parameter updates computed by stochastic gradient descent for each batch will be suboptimal and noisy, a good minimum of the loss can still be found in expectation by iterating over the entire training set. However, as the number of negative documents per query increases to $d+1$, the approximations of the gradients of the loss and thus parameter updates at each training step will be more accurate and therefore training will be more efficient. Increasing the number of negatives to an even higher number is still expected to yield a performance improvement, but at a reduced rate: this is because (a) there is no guarantee that with $d+1$ negatives per query, the positive document will be contained within a simplex of negatives, but with every additional negative this probability increases, and (b) once the positive document is contained within a finite simplex, the probability for every additional negative

[4]Because document vectors are distributed far from the origin, this will typically be true even when the dot product is used as a similarity function.

document to further constraint the bounded subspace where the loss can be minimized becomes increasingly smaller.

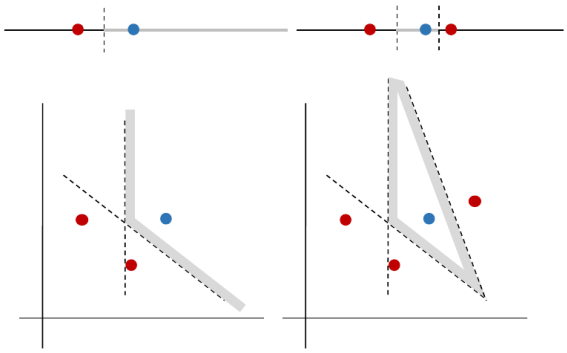

Figure 3: Positive (blue) and negative (red) document vectors in a shared document and query embedding space of 1 (upper row) and 2 (lower row) dimensions. The contrastive learning objective simply requires the query representation to be mapped closer to the positive than all negative documents; this means that for a d-dimensional space, the query representation need not necessarily lie in proximity to the positive document, but simply within an infinite subspace (left column: grey part of line in 1D space, grey-outlined part of plane in 2D space). At least $d + 1$ negative documents are required to constrain the space of favorable query mappings to a bounded convex polytope. Given that the positive document is contained within a simplex formed by $d + 1$ negative documents as vertices in that space (interval in $\mathbb{R}$, triangle in $\mathbb{R}^2$, tetrahedron in $\mathbb{R}^3$ etc), the loss function will favor mapping the query onto another simplex in the same space (right column: grey interval in 1D space, grey-outlined triangular area in 2D space).

The above theoretical analysis can explain the observations made e.g. by Hofstätter et al. (2021), who obtained significantly better performance when increasing the batch size from 32 to 256, or by Qu et al. (2021), who used "cross-batch" random negatives (pooled from different GPUs) to effectively increase the number of "in-batch" negatives to a few thousand documents in order to "reduce the discrepancy between training and inference", and noticed a substantial improvement of performance as a result. Also, in agreement to our explanation above, the performance improvement they observed as a function of the number of negatives quickly saturated when exceeding the dimensionality of the document embedding space.

Finally, it is evident from the analysis above that the closer the negative document representations lie to to the ground truth representation (i.e. the more relevant the negatives are), the smaller the bounded convex subspace will be, a fact which supports the observed importance of challenging negatives in literature (Xiong et al., 2020; Zhan et al., 2021b; Qu et al., 2021; Hofstätter et al., 2021).

A.2 Data

In the present work we only examine passage retrieval, and use the terms "passage" and "document" interchangeably. All data used are in English.

A.2.1 MS MARCO and TREC Deep Learning

Following the standard practice in related contemporary literature, we use the MS MARCO dataset (Bajaj et al., 2018), which has been sourced from open-domain logs of the Bing search engine, for training and evaluating our models. The MS MARCO passage collection contains about 8.8 million documents and the training set contains about 503k queries labeled with one or (rarely) more relevant documents, on a single level of relevance.

For validation we use a subset of 10k samples from "MS MARCO dev", which is a set containing about 56k labeled queries, and refer to it as "MS MARCO dev 10k". As a test set we use a different, officially designated subset of "MS MARCO dev", originally called "MS MARCO dev.small", which contains 6980 queries. However, following standard practice in literature and leaderboards, we refer to it as "MS MARCO dev". We also evaluate on the TREC Deep Learning track 2019 and 2020 test sets, each containing 43 and 54 queries respectively, labeled to an average "depth" of more than 210 document judgements per query, and using 4 levels of relevance: "Not Relevant" (0), "Related" (1), "Highly Relevant" (2) and "Perfect" (3). According to the official (strict) interpretation of relevance labels[5], a level of 1 should not be considered relevant and thus be treated just like a level of 0, while the lenient interpretation considers passages of level 1 relevant when calculating metrics.

A.2.2 TripClick

We additionally evaluate our framework on a different dataset, for two reasons: first, to assess the robustness and generality of our framework across datasets. Second, our approach is premised on using a list-wise loss function while jointly scoring a large number of documents within the same query context, and thus differences from pair-wise approaches are expected to be more pronounced

[5] https://trec.nist.gov/data/deep2019.html

Parameter	Value
Max. query length	32
Max. doc. length (MS MARCO)	256
Max. doc. length (TripClick)	512
Batch size	32
Optimizer	RAdam
Adam epsilon	1.3e-7
Learning rate	1.73e-6
LR warmup steps	9000
Weight decay	9.5e-5
Dropout	0.1
Max. gradient clipping	1.0
d_{model}	768

Table 3: Main configuration parameters of CODER (without transformation of document representations).

when training on a dataset where several documents have been judged with respect to their relevance to a query.

TripClick is a recently introduced health IR dataset (Rekabsaz et al., 2021b) based on click logs that refer to about 1.5M MEDLINE articles. The approx. 700k unique queries in its training set are split into 3 subsets, HEAD, TORSO and TAIL, based on their frequency of occurrence: queries in TAIL are asked only once or a couple of times, while queries in HEAD have been asked tens or hundreds of times. As a result, each query in HEAD, TORSO and TAIL on average ends up with 41.9, 9.1 and 2.8 pseudo-relevance judgements, using a click-through model (RAW) where every clicked document is considered relevant. The dataset also includes alternative relevance judgements using the Document Click-Through Rate (DCTR), on 4 distinct levels (the latter follow the same definitions as the TREC Deep Learning evaluation sets). For validation and evaluation of our models we use the officially designated validation and test set, respectively (3.5k queries each).

A.3 Evaluation

We use mean reciprocal rank (MRR), normalized discounted cumulative gain (nDCG), and recall to evaluate the models on TREC DL tracks, MS MARCO and TripClick, in line with past work (e.g. (Xiong et al., 2020; Zhan et al., 2020a; Hofstätter et al., 2021; Rekabsaz et al., 2021b)). All training and evaluation experiments are produced with the same seed for pseudo-random number generators. While relevance judgements are well-defined in MS MARCO and TripClick, for the TREC DL tracks there exist strict and lenient interpretations of the relevance scores of judged documents (see Section A.2). As past work has been inconsistent in specifying which interpretation is used, we evaluate all models on both versions, denoted as {lenient}/{strict} in Table 1. We calculate the metrics using the official TREC evaluation software.[6]

A.4 Using a transformer as document scoring function

We have experimented with more complex, parametric functions φ as scoring modules, including feedforward neural networks and stacks of transformer blocks (see diagram in Figure 4), which explicitly model inter-document relationships. When we use a transformer encoder as a scoring module, we do not use positional encodings over the input document embeddings, because we require permutation invariance: given fixed positional encodings of ranking order, it would be trivial for the model to learn to assign higher scores for documents of higher rank; at the same time, there is no meaningful sequence order over document embeddings other than relevance ranking.

Although a transformer encoder of 2 blocks was able to marginally outperform the simple function of Equation (2), the small improvement makes it hard to justify precluding the option of single-stage dense retrieval, as well as the additional computational cost. However, in future work, we intend to more thoroughly investigate the use of a transformer-based scoring function, adding more transformer blocks, which we have not been able to implement due to computational constraints - self-attention incurs a $O(N^2)$ GPU memory dependence on the number of context documents N.

A.5 How context can help even without parametric modeling of document relationships

The improvement of our contextual reranking framework over standard triplet training can be attributed to the fact that the model now encounters many more query-specific candidate documents for the same query during a single step of training, which offers a more complete and precise view of the loss landscape and thus leads to a more accurate update of parameters in the same step (also see discussion in Sec. A.1). However, this constitutes a

[6] trec.nist.gov/trec_eval/index.html

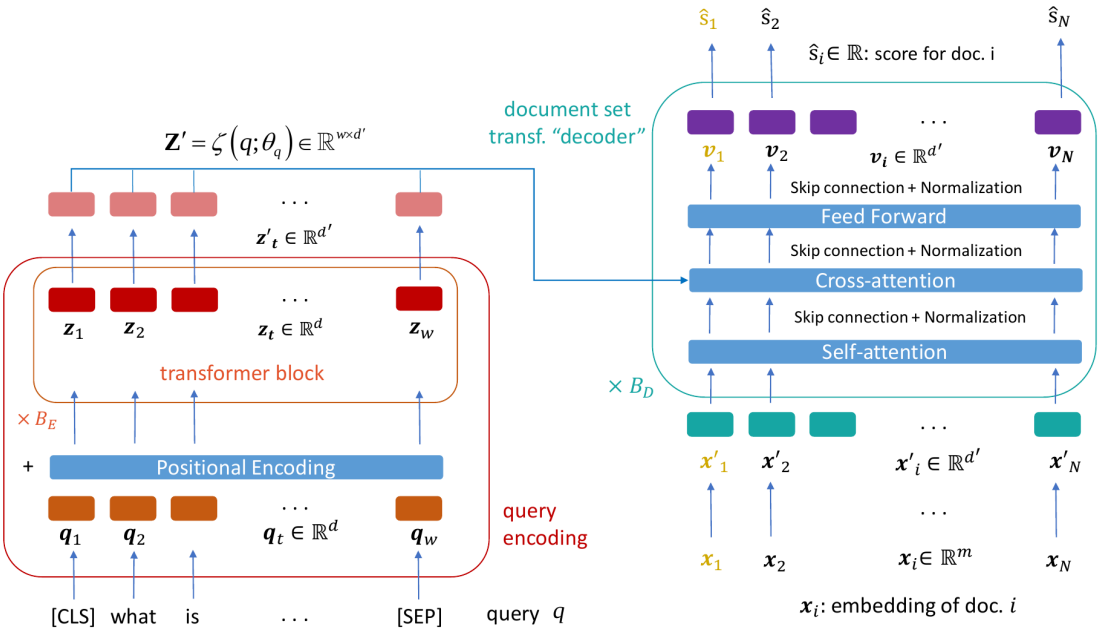

Figure 4: Schematic diagram of CODER employing a transformer "decoder" as a document set scoring module.

"weak" exploitation of ranking context offered by CODER.

The ranking context is exploited more effectively when using a list-wise KL divergence loss. We believe that one reason is that often, among the retrieved candidates, some documents which have not been labeled as positive, are in fact relevant (i.e. false/mislabeled negatives) (Qu et al., 2021). The KL-divergence function is well-positioned to deal with this case, compared to a pair-wise loss (e.g. Max-margin loss) and the Negative Loglikelihood Loss (NLL, a.k.a. InfoNCE) as used e.g. in (Karpukhin et al., 2020) and (Qu et al., 2021). The KL-divergence loss, which compares the distribution of predicted scores against the annotated relevance scores, does not directly penalize assigning a high score to a document annotated as non-relevant; instead, it severely penalizes assigning a low score to a ground-truth relevant document. Therefore, as long as the ground-truth positive document p receives a not-too-low normalized relevance score $\hat{s}_p = \text{softmax}(\varphi(\mathbf{X}))_p$, e.g. $\hat{s}_p > 0.2$, which allows it to escape the very steep part of the loss curve $L(\hat{s}_p) = -\log(\hat{s}_p)$ close to $\hat{s}_p = 0$, the loss will still be small. Thus, it will not severely affect the model's parameters to erroneously force ranking p higher than the false negatives.

Of course, the more such false negative documents exist, receiving non-zero weight in the predicted score distribution, the more difficult it becomes for $\hat{s}_p$ to be high, which leads to a higher loss. However, the existence of more than one labeled positives ($k > 1$) per query alleviates this problem: the individual normalized scores of the k positives will be affected less by the existence of false negatives, and because of the form of $L(x) = -\log(x)$, the overall loss will be smaller than in the case of $k = 1$. This is an additional benefit the KL-divergence loss has over NLL, as the NLL only takes into account a single positive document at a time.

A.6 Notes on RepBERT used for reranking BM25

We surprisingly found RepBERT to be significantly more effective when used as a reranker with BM25 as the first stage retrieval method, although it has been developed and so far only considered and evaluated as a stand-alone dense retrieval method. This is despite RepBERT being an overall much stronger retrieval method than BM25, with a much better recall (0.943 with a cut-off at 1000 passages on MS MARCO dev, versus 0.853 for BM25). This observation may fall within the scope of the general discussion regarding the advantages of combining exact (lexical) and inexact (latent representation) matching (Mitra and Craswell, 2018). One can also hypothesize that BM25 may filter out candidates which would otherwise (spuriously or justi-

Figure 5: Evolution of the loss on the training set, as training of BM25→CODER(RepBERT) progresses. Different curves correspond to a different type and number of documents used as negatives during training. While training loss is decreasing in all cases, only using numerous retrieved candidates as negatives, combined with a list-wise KL-divergence loss, results in a significant improvement of performance over the base method (see Figure 2) on the validation and test sets.

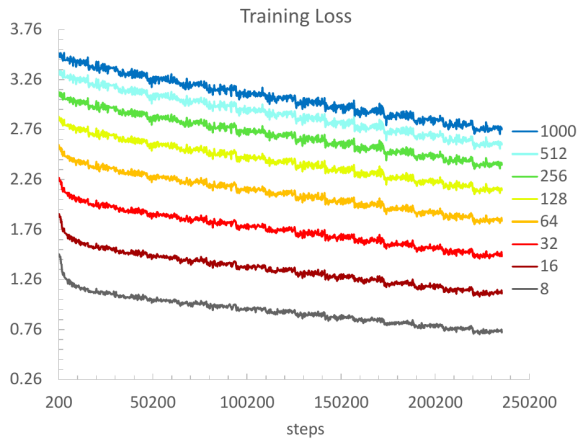

Figure 6: Evolution of the loss on the training set, as training of BM25→CODER(RepBERT) progresses. Different curves correspond to a different number of BM25 candidates used as negatives during training. While training loss is decreasing in all cases, only using a large number of candidates results in a significant improvement of performance over the base method on the validation and test sets (see Figure 2).

fiably) lie close to the ground truth passsage in a latent semantic space, and would have been thus preferred by the deep learning ranker, while not being considered as "hits" in the metrics; this interpretation is supported by the observations of (Qu et al., 2021), who found that very often actually relevant passages have not been labeled as such in MS MARCO.

A.7 Additional Results on TripClick

Table 4 shows the performance of models trained and evaluated on the TripClick dataset. Table 5 shows zero-shot evaluation results for models trained on MS MARCO, but evaluated on TripClick (a dataset of queries and articles in the biomedical domain) without any additional training.

A.8 Detailed comparison with related work

Recent work on ad-hoc information retrieval has employed transformer-based architectures following two main approaches: the first approach (Nogueira and Cho, 2020; Khattab and Zaharia, 2020) allows direct interactions between query and document terms through attention, offering impressive retrieval performance, albeit at the expense of computational efficiency; it can practically be used only as part of a cascade system for reranking candidate documents retrieved by a first-stage method, and still introduces a significant end-to-end processing delay, in the order of seconds per query. These powerful but slow "cross-encoder" models have also been used as teachers for "dual encoder" (a.k.a "bi-encoder") methods (Lin et al., 2020; Qu et al., 2021; Ren et al., 2021a; Hofstätter et al., 2021). These methods constitute the second approach, described below.

The dual encoder approach employs an architecture of two transformer encoders (optionally sharing weights, or implemented as the same encoder, distinguishing between queries and documents by adding special sequence type encodings) to separately encode the query and document sequences, without interactions between them. For inference, it relies on the efficient computation of the dot product through high-performing Approximate Nearest Neighbors libraries such as FAISS (Johnson et al., 2017) to evaluate the similarity between extracted query and document representations. This approach, called "dense retrieval", is highly effective, fast and single-stage, but still lags behind in terms of retrieval performance compared to the first approach.

Our framework addresses the current dilemma of slow reranking versus fast but less effective single-stage retrieval: it efficiently fine-tunes the query encoder of an existing (base) dense retrieval dual encoder model through reranking a set of precomputed document embeddings, offering a substantial

Model	DCTR Head		RAW Head			RAW Torso			RAW Tail		
	MRR	nDCG	MRR	nDCG	Recall	MRR	nDCG	Recall	MRR	nDCG	Recall
BM25[1]	0.276	0.224	-	0.199	0.128	-	0.206	0.262	-	0.267	0.409
Transformer-Kernel[1]	-	0.284	-	0.284	0.167	-	0.272	0.321	-	0.295	0.459
BERT-Dot (SciBERT)[2]	0.530	0.243	-	-	-	-	-	-	-	-	-
BERT-Cat (SciBERT)[2]	0.595	0.294	-	-	-	**0.459**	**0.360**	-	**0.377**	**0.408**	-
RepBERT (abbrev: RB)	0.526	0.255	0.574	0.344	0.199	0.338	0.246	0.309	0.254	0.268	0.404
BM25 → RepBERT	0.538	0.262	0.592	0.356	0.204	0.359	0.269	0.340	0.278	0.297	0.445
RB → CODER(RB, RB)	**0.637***	**0.318***	**0.679***	**0.421***	**0.235***	0.433	0.308	0.355	0.296	0.315	0.469
CODER(RB, RB)	0.634	0.316	0.674	0.419	0.234	0.433	0.308	0.355	0.296	0.315	0.468

Table 4: Performance when applying CODER to RepBERT on the TripClick dataset, using multi-level (DCTR) and binary (RAW) relevance labels (cut-off of 10). The symbol * on best results denotes statistically significant (paired t-test, $p < 0.05$) improvement with respect to all baselines. Results with [1] are from Rekabsaz et al. (2021b), with [2] from Hofstätter et al. (2022).

Model	DCTR Head		RAW Head			RAW Torso			RAW Tail		
	MRR	nDCG	MRR	nDCG	Recall	MRR	nDCG	Recall	MRR	nDCG	Recall
RepBERT-MSM	0.233	0.107	0.278	0.149	0.085	0.205	0.130	0.151	0.117	0.122	0.195
CODER-MSM(RepBERT-MSM)	0.244	0.113	0.294	0.157	0.091	0.211	0.139	0.166	0.127	0.137	0.223
Cocondenser-MSM	0.242	0.114	0.293	0.156	0.091	**0.217**	0.144	0.178	0.153	0.162	0.254
CODER-MSM(Cocondenser-MSM)	**0.251**	**0.117**	**0.305**	**0.161**	**0.093**	0.216	**0.146**	**0.182**	**0.154**	**0.164**	**0.259**

Table 5: Zero-shot results: all models are trained on MS MARCO ('-MSM' suffix) but evaluated on TripClick.

performance improvement over the base model. During inference, it can be used either for rapid reranking in a cascade system, incurring a delay per query of a few milliseconds, or directly for single-stage dense retrieval, at the same speed as the base method, if no non-linear contextual document transformation is employed.

The state of the art has advanced through the exploration of techniques which select better negative documents so as to improve the training process. It has been clearly demonstrated that the quality of negative documents (essentially, how challenging/relevant they are) significantly affects similarity learning and confers performance benefits. Throughout training, Xiong et al. (2020) use the progressively improving document encoder to periodically (every few thousand steps) recompute document representations of all documents in the collection and re-index the document collection using FAISS. They also use the query encoder to recompute all training queries and retrieve the hardest (i.e. highest similarity) negatives through FAISS to construct triplets for the subsequent training period. Zhan et al. (2020a) improve on this effective but very slow and resource-intense process by eschewing fine-tuning of the document encoder, and they instead precompute and fix all document representations; they only fine-tune the query encoder, by retrieving at the end of each training step the hard negatives used for the next training step. Their motivation is two-fold: on the one hand, they argue that performing retrieval through the entire collection at each training step, instead of reranking candidates, reduces the discrepancy between inference retrieval and training tasks, naming their method "Learning to Retrieve" (published as "STAR"/"ADORE" in (Zhan et al., 2021b)). On the other hand, they believe that reranking statically retrieved negatives quickly ceases to be effective, because the model quickly learns to rank static negatives lowly. Although we follow their approach in precomputing document embeddings and fine-tuning only the query encoder, in the present work we show that reranking statically retrieved negatives can indeed be a very effective framework, provided that one establishes a *context* for the query: to achieve this, different from Zhan et al. (2020a), who essentially use dynamically created triplets and a pair-wise loss, we use a large number ($N = 1000$) of static pre-retrieved candidates per query (instead of a single or a couple) in combination with a list-wise loss function. Additionally, when using a parametric scoring function φ, our framework allows transforming document embeddings on the fly based on the context of the other candidate documents and the query itself.

The work of Qu et al. (2021) is also motivated by reducing the discrepancy between training and in-

ference task, and as a solution they increase the quantity of negatives by using a computational framework for parallelism, PaddlePaddle (Yanjun Ma and Yanjun Ma, 2019), to multiply the number of in-batch negatives by the number of GPUs used for training (an approach they call "cross-batch negatives"). As part of an elaborate, multi-step process, they fine-tune a base dual encoder model which has initially been trained through "cross-batch" negatives, after first retrieving the most relevant documents per query to use as static negatives. However, their approach only uses 4 such hard negatives per query, together with a pair-wise negative loglikelihood loss, and relies on several thousands of random cross-batch negatives. It thus differs from ours in that it does not establish a context; in fact, Qu et al. (2021) find that without training a slow but powerful term-interaction model for "denoising" (excluding the most relevant documents from the set of negatives), the performance of the fine-tuned model is substantially worse than the base model. Instead, our framework significantly improves performance when fine-tuning the base model, while being much faster and less demanding in terms of infrastructure. Also, since the number of negatives per query in our case is decoupled from the batch size, it can grow without the possibly detrimental effects that increasing the batch size can have on batch gradient descent optimization.

TAS-B (Hofstätter et al., 2021) goes into the orthogonal direction of improving the quality of negative documents, using a simple but effective idea: it first clusters queries by semantic similarity and then packs queries from the same cluster into the same batch, such that their corresponding ground-truth relevant documents, which are used as in-batch negatives, are no longer random, as documents highly relevant to a query are very likely to also be relevant to the semantically related queries in the same batch. Despite starting from a different motivation, TAS-B can be seen as following an indirect, "noisier" version of our approach: effectively, the scores of several somewhat related in-batch negative documents enter the calculation of the loss for a given query. One difference lies in the type of loss function used: pair-wise in the case of TAS-B, list-wise in our case. This becomes especially important in the case of more than one ground-truth relevant documents (rare in MS MARCO), and/or several levels of relevance judgements. A second, more important difference is that our negatives are the top-$N - k$ documents retrieved by the base retrieval method (where k is the number of positives), and thus for competitive base retrieval models they will almost certainly be of higher relevance than even the related in-batch negatives of TAS-B; at the same time, they are the documents which the model itself considers as the most similar to the query, and thus by definition the most challenging and suitable negatives to be juxtaposed with the ground-truth relevant document.

With respect to simultaneously scoring a set of candidate documents, our approach is reminiscent of list-wise ranking models which employ self-attention (Pang et al., 2020; Pasumarthi et al., 2019b). However, there are several key differences:

- Type of features: to evaluate semantic similarity between query and documents, these methods rely on BM25/TF-IDF interaction between query and document terms. These sparse query-document features, alongside PageRank and features such as freshness and click-through rates, are fed as input vectors to stack of customized self-attention blocks (SetRank, (Pang et al., 2020)) or to a single self-attention layer over document representation followed by concatenation with a query representation and scoring layers (Pasumarthi et al., 2019b).

- Architecture: our framework supports but does not necessarily rely on self-attention or other non-linear transformation of document representations. In the variant where we do make use of self-attention, we do not use positional encoding, in order to enforce permutation invariance. Not only do we represent queries very differently (through a separate query transformer encoder), but training the query encoder is the main objective of our method. Also, query representations can enter as part of a cross-attention module, which allows document representations to interact with query term representations. Thus, our architecture is akin to a complete encoder-decoder transformer (Vaswani et al., 2017): the encoder extracts query *term* representations, while a "decoder" component (without attention masking or positional encoding) concurrently transforms and scores a set of *document embeddings* (see Figure 4 in the Appendix).

- Annotation and learning objective: SetRank uses a custom "attention rank loss function" over provided target rankings, since the datasets used for training (Istella LETOR[7], Microsoft LETOR[8], Yahoo! LETOR[9]) include 46-500 judged documents per query, on 5 levels of relevance.
- They are only used for reranking, with the pool of candidate documents given in the dataset. Our framework allows both reranking retrieved candidates of a first-stage retrieval method, as well as single-stage dense retrieval. During training, one can use dynamic retrieval of negatives through ANN search, following (Zhan et al., 2021b), although we defer investigating this for future work.

In summary, we can say that existing works modeling inter-document relationships and employing list-wise ranking loss functions are specific to the data modalities of "deeply annotated" datasets commonly used for list-wise ranking. As baselines they use models such as LambdaMART (Burges, 2010), RankSVM (Joachims, 2002) and GSF (Ai et al., 2019), and do not consider application of the proposed approaches to sparsely annotated, large scale deep learning datasets such as MS MARCO, nor do they compare or refer to any contemporary transformer-based dense retrieval or reranking models which currently represent the SOTA in ad hoc text retrieval. Likewise, the latter SOTA work does not consider list-wise ranking approaches, or refer and compare to the former line of work. Our present work aspires to bridge the gap between these two relatively isolated research communities by bringing list-wise, context-based ranking methodology to large, pre-trained transformer-based models, used in dense retrieval and trained through sparsely annotated, large scale datasets such as MS MARCO.

A.9 Potential Risks

The neural IR models in these studies, whether in their "base" form or extended by CODER, inherently contain societal biases and stereotypes. As discussed in previous studies (Rekabsaz et al., 2021a; Rekabsaz and Schedl, 2020), these biases originate from already existing biases in the underlying transformer-based language models, as well as the fine-tuning process on the IR collections. Therefore, using these models in practice may lead to unfair treatment of various social groups (e.g. reflected in the representation or order of appearance in ranked lists of retrieval results), and we strongly advocate a conscious and responsible utilization of the models.

However, we note that CODER can potentially help alleviate this problem, as it offers a suitable framework for incorporating algorithmic bias mitigation methods into deep IR models. We have explored and verified this potential in follow-up work.

A.10 Artifacts

All artifacts used for this work (datasets, models, software) are publicly available and the terms of use are available on their respective sources. They have been used within their intended scope of use (research). Artifacts we created based on those may be restricted by terms and conditions specified by the original artifacts (derivatives of data accessed for research purposes should not be used outside of research contexts).

[7] http://blog.istella.it/istella-learning-to-rank-dataset
[8] http://research.microsoft.com/en-us/projects/mslr
[9] http://learningtorankchallenge.yahoo.com