Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval

Luyu Gao and Jamie Callan
Language Technologies Institute
Carnegie Mellon University
{luyug, callan}@cs.cmu.edu

Abstract

Recent research demonstrates the effectiveness of using fine-tuned language models (LM) for dense retrieval. However, dense retrievers are hard to train, typically requiring heavily engineered fine-tuning pipelines to realize their full potential. In this paper, we identify and address two underlying problems of dense retrievers: i) fragility to training data noise and ii) requiring large batches to robustly learn the embedding space. We use the recently proposed Condenser pre-training architecture, which learns to condense information into the dense vector through LM pre-training. On top of it, we propose coCondenser, which adds an unsupervised corpus-level contrastive loss to warm up the passage embedding space. Experiments on MS-MARCO, Natural Question, and Trivia QA datasets show that coCondenser removes the need for heavy data engineering such as augmentation, synthesis, or filtering, and the need for large batch training. It shows comparable performance to RocketQA, a state-of-the-art, heavily engineered system, using simple small batch fine-tuning.[1]

1 Introduction

Building upon the advancements of pre-trained language models (LM; Devlin et al. (2019); Liu et al. (2019)), dense retrieval has become an effective paradigm for text retrieval (Lee et al., 2019; Chang et al., 2020; Karpukhin et al., 2020; Qu et al., 2021; Gao et al., 2021a; Zhan et al., 2022). Recent research has however found that fine-tuning dense retrievers to realize their capacity requires carefully designed fine-tuning techniques. Early works include iterative negative mining (Xiong et al., 2021) and multi-vector representations (Luan et al., 2020). The recent RocketQA system (Qu et al., 2021) significantly improves the performance of a dense retriever by designing an optimized fine-tuning pipeline that includes i) denoising hard negatives, which corrects mislabeling, and ii) large batch training. While this is very effective, the entire pipeline is very heavy in computation and not feasible for people who do not have tremendous hardware resources, like those in academia. In this paper, we ask, instead of directly using the pipeline, can we take the insights of RocketQA to perform language model pre-training such that the pre-trained model can be easily fine-tuned on any target query set.

Concretely, we ask what the optimized training in RocketQA solves. We hypothesize that typical LMs are sensitive to mislabeling, which can cause detrimental updates to the model weights. Denoising can effectively remove the bad samples and their updates. On the other hand, for most LMs, the CLS vectors are either trained with a simple task (Devlin et al., 2019) or not explicitly trained at all (Liu et al., 2019). These vectors are far from being able to form an embedding space of passages (Lee et al., 2019). The large training batches in RocketQA help the LM to stably learn to form the full embedding space. To this end, we want to pre-train an LM such that it is locally noise-resistant and has a well-structured global embedding space. For noise resistance, we borrow the Condenser pre-training architecture (Gao and Callan, 2021), which performs language model pre-training actively conditioning on the CLS vector. It produces an information-rich CLS representation that can robustly condense an input sequence. We then introduce a simple corpus level contrastive learning objective: given a target corpus of documents to retrieve from, at each training step sample text span pairs from a batch of documents and train the model such that the CLS embeddings of two spans from the same document are close and spans from different documents are far apart. Combining the two, we propose coCondenser pre-training, which unsupervisedly learns a corpus-aware pre-trained model for dense retrieval.

[1] Our code is available at https://github.com/luyug/Condenser.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2843 - 2853

In this paper, we test coCondenser pre-training on two popular corpora, Wikipedia and MS-MARCO. Both have served as information sources for a wide range of tasks. This popularity justifies pre-training models specifically for each of them. We directly fine-tune the pre-trained coCondenser using small training batches without data engineering. On Natural Question, TriviaQA, and MS-MARCO passage ranking tasks, we found that the resulting models perform on-par or better than RocketQA and other contemporary methods.

2 Related Work

Dense Retrieval Transformer LM has advanced the state-of-the-art of many NLP tasks (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Lan et al., 2020) including dense retrieval. Lee et al. (2019) are among the first to demonstrate the effectiveness of Transformer dense retrievers. They proposed a simple Inverse Cloze Task (ICT) method to further pre-train BERT (Devlin et al., 2019). Follow-up works explored other pre-training tasks (Chang et al., 2020) as well end-to-end co-training of reader and retriever (Guu et al., 2020). Karpukhin et al. (2020) is the first to discover that careful fine-tuning can learn effective dense retriever directly from BERT. Later works then started to investigate ways to further improve fine-tuning (Xiong et al., 2021; Qu et al., 2021). Among them, Qu et al. (2021) proposed the RocketQA fine-tuning pipeline which hugely advanced the performance of dense retrievers.

Until very recently, pre-training for dense retrieval has been left unexplored. A concurrent work, DPR-PAQ (Oğuz et al., 2021), revisits pre-training and proposes domain matched pre-training, using a 65-million-size synthetic QA pair dataset generated with pre-trained Natural Question and Trivia QA pipelines to pre-train dense retrievers.

This paper uses a recently proposed dense retrieval pre-training architecture, Condenser (Gao and Callan, 2021). Unlike previous works that design pre-training tasks, Condenser explored the idea of designing a special pre-training architecture to improve representation effectiveness.

One reason why dense retrieval is of immediate great value is that there is a rich literature that studies efficient dense retrieval for first stage retrieval (Johnson et al., 2017; Guo et al., 2020). There are also mature dense retrieval libraries, such as FAISS (Johnson et al., 2017). By pre-encoding the corpus into a MIPS index, retrieval can run online with millisecond-level latency (Johnson et al., 2017; Guo et al., 2020).

Contrastive Learning Contrastive learning has become a very popular topic in computer vision (Chen et al., 2020; He et al., 2020). Recent works have brought the idea to natural language processing to learn high-quality sentence representation (Giorgi et al., 2020; Wu et al., 2020). In this work, we use contrastive learning to do pre-training for dense retrieval. Different from earlier work, instead of individual representations (Giorgi et al., 2020), we are interested in the full learned embedding space, which we will use to warm start the retriever.

The large batch requirement had been a limiting factor in contrastive learning (Chen et al., 2020) under resource-limited setups where GPU (accelerator) memory is not sufficiently large. In general, this extends to any training procedure that uses contrastive loss, including dense retrieval pre-training (Guu et al., 2020; Chang et al., 2020). Gao et al. (2021b) recently devised a gradient cache technique that upper-bounds peak memory usage of contrastive learning to almost constant. In subsection 3.3, we show how to adapt it for coCondenser pre-training.

3 Method

In this section, we first give a brief review of Condenser. Then we discuss how to extend it to coCondenser and how to perform memory-efficient coCondenser pre-training.

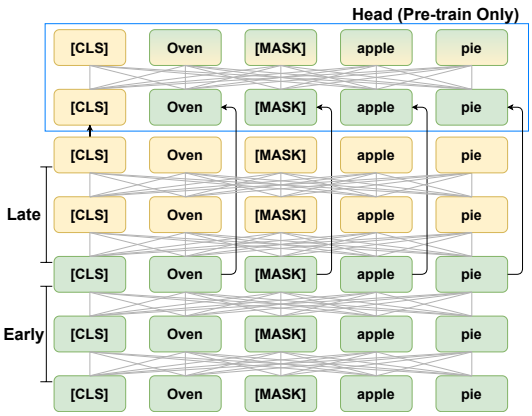


Figure 1: Condenser: Shown are 2 early and 2 late backbone layers. In our experiments each have 6 layers. Condenser Head is dropped during fine-tuning. This illustration is taken from the Condenser paper (Gao and Callan, 2021).

3.1 Condenser

In this paper, we adopt a special pre-training architecture, Condenser (Gao and Callan, 2021). Condenser is a stack of Transformer blocks. As shown in Figure 1, these Transformer blocks are divided into three groups, early backbone encoder layers, late backbone encoder layers, and head layers. An input $x = [x_1, x_2, ..]$ is first prepended a CLS, embedding, and run through the backbone layers.

$$[h_{cls}^0; h^0] = \text{Embed}([\text{CLS}; x]) \quad (1)$$

$$[h_{cls}^{early}; h^{early}] = \text{Encoder}_{\text{early}}([h_{cls}^0; h^0]) \quad (2)$$

$$[h_{cls}^{late}; h^{late}] = \text{Encoder}_{\text{late}}([h_{cls}^{early}; h^{early}]) \quad (3)$$

The head takes the CLS representation from the late layers but using a short circuit, the token representations from the early layers. This late-early pair then runs through the head's Transformer blocks.

$$[h_{cls}^{cd}; h^{cd}] = \text{Head}([h_{cls}^{late}; h^{early}]) \quad (4)$$

The head's outputs are then used to do masked language model (MLM; Devlin et al. (2019)) training.

$$\mathcal{L}^{\text{mlm}} = \sum_{i \in \text{masked}} \text{CrossEntropy}(W h_i^{cd}, x_i) \quad (5)$$

To utilize the capacity of the late layers, Condenser is forced to learn to aggregate information into the CLS token, which will then participate in the LM prediction. Leveraging the rich and effective training signal produced by MLM, Condenser learns to utilize the powerful Transformer architecture to generate dense CLS representations. We hypothesize that with this LM objective typically used to train token representations now put on the dense CLS representation, the learned LM gains improved robustness against noise.

3.2 coCondenser

While Condenser can be trained on a diverse collection of corpora to produce a universal model, it is not able to solve the embedding space issue: while information embedded in the CLS can be non-linearly interpreted by the head, inner products between these vectors still lack semantics. Consequently, they do not form an effective embedding space. To this end, we augment the Condenser MLM loss with a contrastive loss. Unlike previous work that pre-trains on artificial query passage pairs, in this paper, we propose to simply pre-train the passage embedding space in a query-agnostic fashion, using a contrastive loss defined over the target search corpus. Concretely, given a random list of n documents $[d_1, d_2, ..., d_n]$, we extract randomly from each a pair of spans, $[s_{11}, s_{12}, ..., s_{n1}, s_{n2}]$. These spans then form a training batch of coCondenser. Given a span s_{ij}'s corresponding *late* CLS representation h_{ij}, its corpus-aware contrastive loss is defined over the batch as shown below.

$$\mathcal{L}_{ij}^{co} = -\log \frac{\exp(\langle h_{i1}, h_{i2} \rangle)}{\sum_{k=1}^{n} \sum_{l=1}^{2} \mathbb{I}_{ij \neq kl} \exp(\langle h_{ij}, h_{kl} \rangle)} \quad (6)$$

Familiar readers may recognize this as the contrastive loss from SimCLR (Chen et al., 2020), for which we use random span sampling as augmentation. Others may see a connection to noise contrastive estimation (NCE). Here we provide an NCE narrative. Following the spirit of the distributional hypothesis, passages close together should have similar representations while those in different documents should have different representations. Here we use random spans as surrogates of passages and enforce the distributional hypothesis through NCE, as word embedding learning in Word2Vec (Mikolov et al., 2013). We can also recognize this as a span-level language model objective, or "skip-span". Denote span s_{ij}'s Condenser MLM loss $\mathcal{L}_{ij}^{\text{mlm}}$. The batch's loss is defined as an average sum of MLM and contrastive loss, or from an alternative perspective, word and span LM loss.

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{2} [\mathcal{L}_{ij}^{\text{mlm}} + \mathcal{L}_{ij}^{co}] \quad (7)$$

3.3 Memory Efficient Pre-training

The RocketQA pipeline uses supervision and large-batch training to learn the embedding space. We would also like to run large-batch unsupervised pre-training to construct effective stochastic gradient estimators for the contrastive loss in Equation 6. To remind our readers, this large-batch pre-training happens only once for the target search corpus. We will show that this allows effective small batch fine-tuning on task query sets.

However, due to the batch-wise dependency of the contrastive loss, it requires fitting the large batch into GPU (accelerator) memory. While this can be done naively with interconnected GPU nodes or TPU pods, which can have thousands of gigabytes of memory, academia, and smaller organizations are often restricted to machines with

four commercial GPUs. To break the memory constraint and perform effective contrastive learning, we adjust the gradient caching technique (Gao et al., 2021b) for our setup. We describe the procedure here for people who want to perform coCondenser pre-training but have limited resources. Denote $\mathcal{L}^{co} = \sum_i \sum_j \mathcal{L}^{co}_{ij}$, we can write Equation 7 as,

$$\mathcal{L} = \frac{1}{2n}[\mathcal{L}^{co} + \sum_i \sum_j \mathcal{L}^{\text{mlm}}_{ij}] \tag{8}$$

The spirit of gradient caching is to decouple representation gradient and encoder gradient computation. Before computing the model weight update, we first run an extra backbone forward for the entire batch. This provides numerical values of $[h_{11}, h_{12}, ..., h_{n1}, h_{n2}]$, from which we compute:

$$v_{ij} = \frac{\partial}{\partial h_{ij}} \sum_i \sum_j \mathcal{L}^{co}_{ij} = \frac{\partial \mathcal{L}^{co}}{\partial h_{ij}} \tag{9}$$

i.e. the contrastive loss gradient with respect to the CLS vector. We store all these vectors in a gradient cache, $C = [v_{11}, v_{12}, .., v_{n1}, v_{n2}]$. Using v_{ij}, denote the model parameter Θ, we can write the derivative of the contrastive loss as shown below.

$$\frac{\partial \mathcal{L}^{co}}{\partial \Theta} = \sum_i \sum_j \frac{\partial \mathcal{L}^{co}}{\partial h_{ij}} \frac{\partial h_{ij}}{\partial \Theta} \tag{10}$$

$$= \sum_i \sum_j v_{ij}^\top \frac{\partial h_{ij}}{\partial \Theta} \tag{11}$$

We can then write the gradient of Equation 8.

$$\frac{\partial \mathcal{L}}{\partial \Theta} = \frac{1}{2n} \sum_i \sum_j [v_{ij}^\top \frac{\partial h_{ij}}{\partial \Theta} + \frac{\partial \mathcal{L}^{\text{mlm}}_{ij}}{\partial \Theta}] \tag{12}$$

Since v_{ij} is already in the cache C, each summation term now only concerns span s_{ij} and its activation, meaning that we can compute the full batch's gradient in an accumulation fashion over small sub-batches. In other words, the full batch no longer needs to concurrently reside on the GPUs.

3.4 Fine-tuning

At the end of pre-training, we discard the Condenser head, keeping only the backbone layers. Consequently, the model reduces to its backbone, or effectively a Transformer Encoder. We use the backbone weights to initialize query encoder f_q and passage encoder f_p. Each outputs the last layer CLS. Recall that they have already been warmed up in pre-training. A (query q, passage p) pair similarity is defined as an inner product.

$$s(q, p) = \langle f_q(q), f_p(p) \rangle \tag{13}$$

Query and passage encoders are supervisedly fine-tuned on the target task's training set. We train with a supervised contrastive loss and compute for query q, negative log likelihood of a positive document d^+ against a set of negatives $\{d_1^-, d_2^-, ..d_l^-..\}$.

$$\mathcal{L} = -\log \frac{\exp(s(q, d^+))}{\exp(s(q, d^+)) + \sum_l \exp(s(q, d_l^-))} \tag{14}$$

We run a two-stage training as described in the DPR (Karpukhin et al., 2020) toolkit. As shown in Figure 2b, in the first stage, the retrievers are trained with BM25 negatives. The first-stage retriever is then used to mine hard negatives to complement the negative pool. The second stage retriever trains with the negative pool generated in the first round. This is in contrast to the multi-stage pipeline of RocketQA shown in Figure 2a.

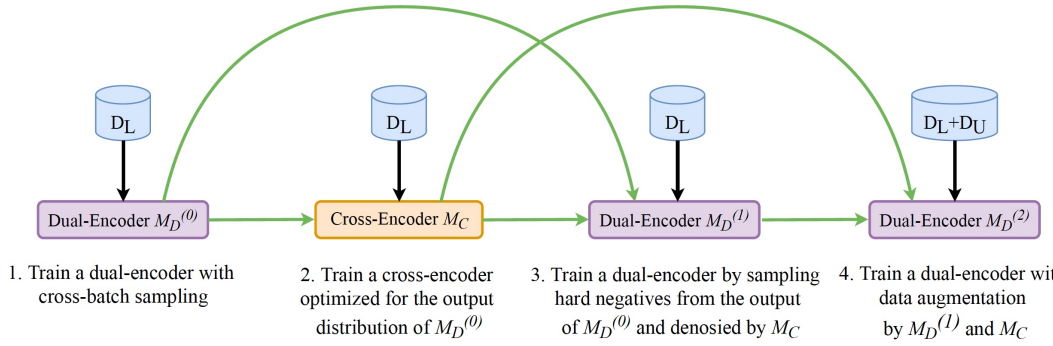
4 Experiments

In this section, we first describe the implementation details of coCondenser pre-training. We then conduct dense retrieval experiments to test the effectiveness of fine-tuned coCondenser retrievers.

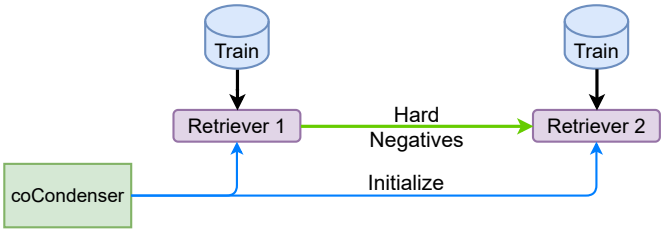
4.1 Pre-training

The coCondenser pre-training starts with vanilla BERT and goes in two stages, universal Condenser pre-training and corpus aware coCondenser pre-training. In the first stage, we pre-train a Condenser and warm start the backbone layers with pre-trained 12-layer $\text{BERT}_{\text{base}}$ weights (Devlin et al., 2019). The backbone uses an equal split, 6 early layers, and 6 late layers. The Condenser pre-training uses the same data as BERT: English Wikipedia and the BookCorpus. The first stage Condenser pre-training takes roughly a week on 4 RTX 2080 Ti GPUs or 2 days on a v3-8 cloud TPU.

The Condenser model from stage one, including both backbone and head, is taken to warm start stage two coCondenser pre-training on the target corpus (Wikipedia or MS-MARCO web collection). We keep the Condenser architecture unchanged in the second step. We use AdamW optimizer with a learning rate 1e-4, weight decay of 0.01, and linear learning rate decay. Each model weight update uses 2K documents. We train using gradient



(a) RocketQA retriever training pipeline (taken from Qu et al. (2021)).



(b) coCondenser retriever training pipeline.

Figure 2: RocketQA training pipelines and two-round retriever training pipeline in coCondenser.

cache update, as described in subsection 3.3. We used the released Condenser model for the first stage. The second stage takes roughly 2 days on 4 RTX 2080 Ti GPUs or 19 hours on a v3-8 cloud TPU. Our GPU implementations are based on Pytorch (Paszke et al., 2019) and TPU implementations on JAX (Bradbury et al., 2018).

After the second stage finishes, we discard the Condenser head, resulting in a model of the *exact* same architecture as $\text{BERT}_{\text{base}}$.

4.2 Dense Passage Retrieval

Next, we fine-tune the learned coCondenser to test retrieval performance. Following RocketQA, we test on Natural Question and MS-MARCO passage ranking. We also report performance on Trivia QA, whose pre-processed version is released in DPR.

4.2.1 Setup

Dataset We use MS-MARCO passage ranking (Bajaj et al., 2018), Natural Question(NQ; Kwiatkowski et al. (2019)) and Trivia QA(TQA; Joshi et al. (2017)). MS-MARCO is constructed from Bing's search query logs and web documents retrieved by Bing. Natural Question contains questions from Google search. Trivia QA contains a set of trivia questions. We report official metrics MRR@10, Recall@1000 for MS-MARCO, and Recall at 5, 20, and 100 for NQ and TQA.

Data Preparation We use Natural Question, Trivia QA, and Wikipedia cleaned and released with DPR toolkit (Karpukhin et al., 2020). NQ and TQA each have about 60K training data post-processing. Similarly, we use the MS-MARCO corpus released with RocketQA open-source code (Qu et al., 2021). For reproducibility, we use the official relevance file instead of RocketQA's extended one, which has about 0.5M training queries. The BM25 negatives for MS-MARCO are taken from the official training triples.

Training MS-MARCO models are trained with Tevatron toolkit (Gao et al., 2022) using AdamW with a 5e-6 learning rate, linear learning rate schedule, and batch size 64 for 3 epochs. NQ and TQA models are trained with the DPR toolkit following published hyperparameters by Karpukhin et al. (2020). All models are trained on one RTX 2080 Ti. We added gradient caching to DPR to deal with memory constraints. The models are trained only on each task's corresponding training set. We note that RocketQA is trained on a concatenation of several datasets (Qu et al., 2021).

Model Validation Since for dense retrieval, validating a checkpoint requires encoding the full corpus, evaluating a checkpoint becomes very costly. Due to our computation resource limitation, we follow the suggestion in the DPR toolkit and take

Method	MS-MARCO Dev		Natural Question Test			Trivia QA Test		
	MRR@10	R@1000	R@5	R@20	R@100	R@5	R@20	R@100
BM25	18.7	85.7	-	59.1	73.7	-	66.9	76.7
DeepCT	24.3	90.9	-	-	-	-	-	-
docT5query	27.7	94.7	-	-	-	-	-	-
GAR	-	-	60.9	74.4	85.3	73.1	80.4	85.7
DPR	-	-	-	74.4	85.3	-	79.3	84.9
ANCE	33.0	95.9	-	81.9	87.5	-	80.3	85.3
ME-BERT	33.8	-	-	-	-	-	-	-
RocketQA	37.0	97.9	74.0	82.7	88.5	-	-	-
Condenser	36.6	97.4	-	83.2	88.4	-	81.9	86.2
DPR-PAQ								
- $\text{BERT}_{\text{base}}$	31.4	-	74.5	83.7	88.6	-	-	-
- $\text{BERT}_{\text{large}}$	31.1	-	75.3	84.4	88.9	-	-	-
- $\text{RoBERTa}_{\text{base}}$	32.3	-	74.2	84.0	**89.2**	-	-	-
- $\text{RoBERTa}_{\text{large}}$	34.0	-	**76.9**	**84.7**	**89.2**	-	-	-
coCondenser	**38.2**	**98.4**	**75.8**	**84.3**	89.0	**76.8**	**83.2**	**87.3**

Table 1: Retrieval performance on MSMARCO dev, Natural Question test and Trivia QA test. We mark bold the best performing models as well as the best performing 12-layer base models. Results unavailable are left blank.

the last model training checkpoint. We do the same for MS-MARCO.

Comparison Systems We take RocketQA (Qu et al., 2021), the state-of-the-art fine-tuning technique, as our main baseline.

We borrowed several other baselines from the RocketQA paper, including lexical systems BM25, DeepCT (Dai and Callan, 2019), DocT5Query (Nogueira and Lin, 2019) and GAR (Mao et al., 2020); and dense systems DPR (Karpukhin et al., 2020), ANCE (Xiong et al., 2021), and ME-BERT (Luan et al., 2020).

We also included the concurrent work DPR-PAQ (Oğuz et al., 2021), which pre-trains using a 65-million-size synthetic QA pair dataset. The pre-training data is created by using retriever-reader pairs trained on Natural Question and Trivia QA. Designing the synthesis procedure also requires domain knowledge, thus under the context of this paper, we refer to this as a *semi-supervised* pre-training method. We include 4 DPR-PAQ variants based on base/large architectures of BERT/RoBERTa models.

Finally, we fine-tune a Condenser model which is produced in the first stage of pre-training.

4.2.2 Results

Table 1 shows development (dev) set performance for MS-MARCO passage ranking and test set performance for Natural Question and Trivia QA. Across three query sets, dense systems show superior performance compared to sparse systems. We also see a big performance margin between systems involving either careful fine-tuning or pre-training (RocketQA, DPR-PAQ, Condenser, coCondenser) over earlier dense systems. This result confirms recent findings that low dimension embeddings possess a strong capacity for dense retrieval, a capacity however hard to exploit naively.

coCondenser shows small improvements over RocketQA. Importantly, this is achieved with *greatly* reduced computation and data engineering effort in fine-tuning. Notably on MS-MARCO, coCondenser reduced the RocketQA's 4096 batch size to 64 (Table 5). A comparison of the two training pipelines of RocketQA and coCondenser can be found in Figure 2.

Comparison with DPR-PAQ shows several interesting findings. Combining large semi-supervised pre-training with the better and larger LM $\text{RoBERTa}_{\text{large}}$, DPR-PAQ achieves the best results on Natural Question. On the other hand, when starting from BERT (base/large), DPR-PAQ shows similar performance to coCondenser, which is based on $\text{BERT}_{\text{base}}$. This suggests that large-scale semi-supervised pre-training is still the way to go to get the very best performance. However, when computational resources are limited and a large pre-training set is missing, the unsupervised coCondenser is a strong alternative. On the other hand, as it moves to MS-MARCO where DPR-PAQ's pre-training supervision becomes distant,

MS-MARCO Passage Ranking Leaderboard		
Rank	Method	**MRR@10**
1	Adaptive Batch Scheduling + CoCondenser	43.1
2	coCondenser*	42.8
MS-MARCO Document Ranking Leaderboard		
Rank	Method	**MRR@100**
1	UniRetriever	44.0
2	coCondenser + MORES+*	43.6

Table 2: Performance on the MS-MARCO leaderboards with reranking. *Our submissions.

we observe that DPR-PAQ becomes less effective than RocketQA and coCondenser.

The comparison between Condenser and coCondenser demonstrates the importance of the contrastive loss in coCondener: coCondenser can be robustly fine-tuned thanks to its pre-structured embedding space, allowing it to have better Recall (fewer false negatives) across all datasets.

4.3 Reranking on MS-MARCO Eval

Due to the leaderboard nature of MS-MARCO Eval set, we cannot do ablation studies on it but have only made two submissions. We follow other top-performing systems and add some form of reranker. For the passage ranking leaderboard, we rerank the top 1000 retrieved passages with an ensemble of ERNIE (Sun et al., 2020) and RoBERTa. We also fine-tuned a coCondenser on the MS-MARCO document ranking dataset. As passage retrieval is the focus of this paper, we retrieve based on the first passage of 512 tokens. The top100 are reranked by a fast modular reranker (Gao et al., 2020). Performance of best systems and ours are recorded in Table 2. At the time of this paper's submission, both of our systems are the 2nd best on the two leaderboards. For passages ranking, we are excited to see that other people are able to further improve coCondenser with additional fine-tuning techniques. For document, we leave the study of retrieval beyond first passage to future work and refer readers to other leaderboard systems.

5 Analysis

Recall the two desired properties of coCondernser are local noise resistance to mislabeling and a well-structured, pre-trained embedding space. In this section, to investigate the former, we introduce and compare with a knowledge distillation (Hinton et al., 2015) setup where we substitute noisy hard labels with soft labels from a cross-encoder. For the latter, we measure the quality of 1st stage retriever mined negatives to see if coCondenser embedding can help find related but not relevant hard negatives. We also provide ablation studies of loss components and, pre-training and fine-tuning stages.

5.1 Learning with Soft Labels

To analyze the local robustness of coCondenser, we introduce a knowledge distillation upper-bound model: instead of training using the noisy labels, we first train a cross encoder and then fine-tune a coCondenser model using soft labels generated from the cross-encoder. Unlike Qu et al. (2021) that uses cross encoder for filtering, here we directly expose the logits as soft labels. Concretely, given cross encoder g, a batch of M queries $\{q_1, q_2, .., q_M\}$, each paired with N passages (positive and hard negatives) $\{p_{11}, .., p_{1N}, .., p_{MN}\}$, for a query q_l we define its soft target distribution T,

$$T_{ij} = \text{softmax}_j(g(q_l, d_{ij})) \text{ if } i = l \text{ else } 0 \quad (15)$$

i.e., the soft labels are normalized logits from g for the local passages and 0 for the rest. Let $S_{ij} = \text{softmax}_{ij}(s(q, p_{ij}))$, the normalized bi-encoder similarities. Loss is defined as Kullback-Leibler divergence between S and T,

$$\mathcal{L}_{kd} = D_{\text{KL}}(S||T) \quad (16)$$

This setup a) focuses on improving labels for *local* hard negatives and positives while b) avoids evaluating cross encoder g for in-batch negatives. In Table 3, we compare coCondenser trained with

		MS-MARCO Dev		
Model	Label	MRR@10	R@100	R@1K
BERT	Hard	33.4	85.1	95.4
coCondenser	Hard	38.2	91.3	98.4
coCondenser	Soft	**39.1**	**91.9**	**98.6**

Table 3: Fine-tuning with hard v.s. soft labels.

the original hard labels and the cross-encoder generated soft labels. Using soft labels indeed produces some improvement. On the other hand, without help from the cross-encoder, coCondenser still yields performance within small margins, showing coCondenser's superior local noise resistance.

5.2 Quality of Mined Negatives

Intuitively, a globally better-structured embedding space will be less likely to collapse, producing a more accurate set of mined negatives for the second stage retriever to learn over. In other words, the

2nd stage retriever will produce less unexpected top (hard) negatives. To quantitatively measure this, we propose a new metric, top n neighborhood recall at depth k (nb-recall^n@k): for a query, the coverage over the 2nd stage retriever's top n candidates by the 1st stage retrievers top k candidates,

$$\text{nb-recall}^n@k = \frac{\{\text{stage1 top k}\} \cap \{\text{stage2 top n}\}}{n} \quad (17)$$

Essentially, this measures how the mined hard negatives agree with the actual negatives, or simply, how well the 1st stage retriever locating hard negatives. We measure neighborhood recall of BERT, Condenser and coCondenser retrievers, averaged over all MS-MARCO Dev queries, for $n = 50, 100$ and various k values, in Figure 3, We see consistent higher nb-recall of Condenser over BERT and coCondenser over Condenser. The former comes from stronger CLS representation while the latter is due to the globally better-structured embedding space.

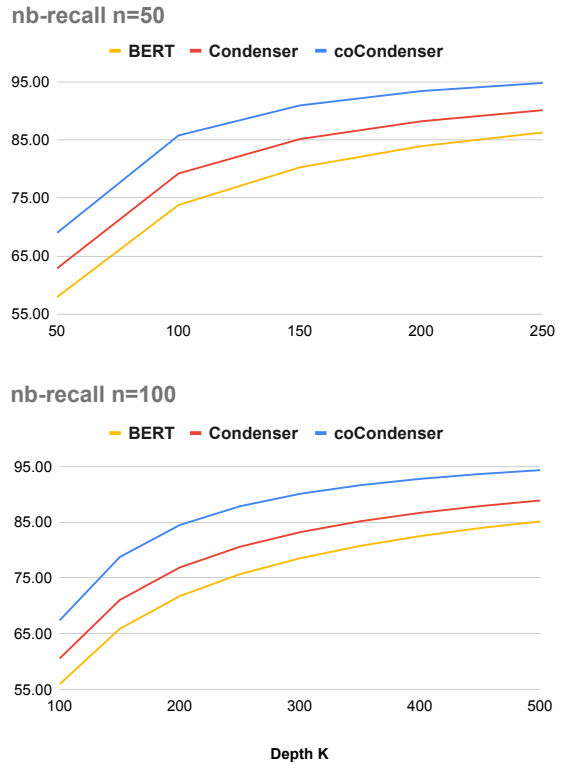


Figure 3: MS-MARCO Dev Neighborhood Recall

5.3 Pre-training Loss Ablation

In this section, we conduct an ablation study to understand the second stage pre-training loss components' influence on the final quality. In particular, we consider a Condenser model further pre-trained with only the contrastive loss.

	MS-MARCO Dev		
Model	MRR@10	R@100	R@1K
Condenser	36.6	89.4	97.4
+ Contrative loss	36.7	90.3	98.0
+ MLM loss†	**38.2**	**91.3**	**98.4**

Table 4: Effect of loss components. †: This is the coCondenser model.

In Table 4, we see that further pre-training Condenser with only contrastive loss leads to better recall but similar MRR. The contrastive loss learns a better embedding space but by itself cannot keep the CLS locally discriminative. The original Condenser is able to rank better locally, producing similar MRR with fewer recalled passages. When both contrastive and Condenser MLM loss are used, we see improvements on all metrics. This again stresses the importance of the Condenser MLM loss during the second contrastive learning stage.

5.4 Training Stages

We seek to understand the contribution of each pre-training and fine-tuning stage of coCondenser retriever. We consider pre-trained Condenser from the first stage and coCondenser from the second stage. For each, we consider retrievers trained with and without hard negatives (HN). For reference, we compare also with various RocketQA training stages. Results are shown in Table 5. We see that each stage of RocketQA is critical. As each is added, performance improves steadily. On the other hand, this also suggests the full pipeline has to be executed to get the best performance. In comparison, we see Condenser with hard negatives has

Method	**Batch Size**	**MS-MARCO Dev**	
		MRR@10	R@1000
RocketQA			
Cross-batch negatives	8192	33.3	-
+ Hard negatives	4096	26.0	-
+ Denoising	4096	36.4	-
+ Data augmentation	4096	37.0	97.9
coCondenser			
Condenser w/o HN	64	33.8	96.1
+ Hard negatives	64	36.6	97.4
coCondenser w/o HN	64	35.7	97.8
+ Hard negatives	64	**38.2**	**98.4**

Table 5: MS-MARCO Dev performance for various training stages RocketQA and coCondenser.

performance very close to the full RocketQA system. Condenser with hard negatives also has better MRR than coCondenser without hard negatives, meaning that Condenser from the first pre-training stage is already very strong locally but the embedding space trained from a relatively cold start is still not optimal, causing global misses. Adding the corpus aware loss, coCondenser without hard negatives has Recall very close to the full RocketQA system, using only a size 64 batch. This again confirms our hypothesis that fine-tuning can benefit from a pre-trained passage embedding space. Further adding hard negatives, we get the strongest coCondenser system that is both locally and globally effective. Note that all Condenser systems achieve their performance *without* denoising, showing the superior noise resistance capability learned using the Condenser architecture. Practically, our systems also do not require data augmentation, which removes engineering effort in designing augmentation techniques and defining augmentation data.

6 Conclusion

This paper introduces *coCondenser*, an unsupervised corpus-aware language model pre-training method. We demonstrate proper pre-training can establish not only language understanding ability but also corpus-level representation power. Leveraging the Condenser architecture and a corpus aware contrastive loss, coCondenser acquires two important properties for dense retrieval, noise resistance, and structured embedding space. This corpus-aware pre-training needs to be done once for a search corpus and is query agnostic. The learned model can be shared among various types of end task queries.

Experiments show that coCondenser can drastically reduce the costs of fine-tuning a strong dense retriever. We also find coCondenser yields performance close or similar to semi-supervised pre-trained models that are several times larger.

Importantly, coCondenser provides a *hands-off* way to pre-train a very effective LM for dense retrieval. In particular, it effectively removes the effort for designing and testing pre-training as well as fine-tuning techniques. With our models, practitioners can use limited resources to train dense retrieval systems with state-of-the-art level performance. Future works may also investigate integrating additional pre-training and/or fine-tuning methods to further improve performance.

Acknowledgments

This research was supported by NSF grant IIS-1815528. Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors. The authors would like to thank Google's TPU Research Cloud (TRC) for access to Cloud TPUs and the anonymous reviewers for the reviews.

References

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.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709.

Zhuyun Dai and J. Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *ArXiv*, abs/1910.10687.

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

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

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Modularized transfomer-based ranking framework. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4180–4190, Online. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021a. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*.

Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval. *ArXiv*, abs/2203.05765.

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

John Michael Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. 2020. Declutr: Deep contrastive learning for unsupervised textual representations. *ArXiv*, abs/2006.03659.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*.

Kelvin Guu, Kenton Lee, Z. Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735.

Geoffrey E. Hinton, Oriol Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

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

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

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

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

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Y. Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *ArXiv*, abs/2005.00181.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020. Generation-augmented retrieval for open-domain question answering.

Tomas Mikolov, Kai Chen, G. S. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to doctttttquery.

Barlas Oğuz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen tau Yih, Sonal Gupta, and Yashar Mehdad. 2021. Domain-matched pre-training tasks for dense retrieval.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

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

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975.

Z. Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *ArXiv*, abs/2012.15466.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Z. Yang, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning discrete representations via constrained clustering for effective and efficient dense retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1328–1336.