Condenser: a Pre-training Architecture for Dense Retrieval

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

Abstract

Pre-trained Transformer language models (LM) have become go-to text representation encoders. Prior research fine-tunes deep LMs to encode text sequences such as sentences and passages into single dense vector representations for efficient text comparison and retrieval. However, dense encoders require a lot of data and sophisticated techniques to effectively train and suffer in low data situations. This paper finds a key reason is that standard LMs' internal attention structure is not ready-to-use for dense encoders, which needs to aggregate text information into the dense representation. We propose to pre-train towards dense encoder with a novel Transformer architecture, Condenser, where LM prediction CONditions on DENSE Representation. Our experiments show Condenser improves over standard LM by large margins on various text retrieval and similarity tasks.[1]

1 Introduction

Language model (LM) pre-training has been very effective in learning text encoders that can be fine-tuned for many downstream tasks (Peters et al., 2018; Devlin et al., 2019). Deep bidirectional Transformer encoder (Vaswani et al., 2017) LMs like BERT (Devlin et al., 2019) are the state-of-the-art. Recent works fine-tune the CLS token to encode input text sequence into a single vector representation (Lee et al., 2019; Chang et al., 2020; Karpukhin et al., 2020). The resulting model is referred to as dense encoder or bi-encoder. Fine-tuning associates with vector similarities some practical semantics, e.g., textual similarity or relevance, and therefore the vectors can be used for efficient text comparison or retrieval by inner product. Despite their efficiency, bi-encoders are hard to train. Even with sufficient data, bi-encoders still require carefully designed sophisticated methods to train effectively (Xiong et al., 2021; Qu et al., 2020; Lin et al., 2020). They can also take big performance hits in low data situations (Karpukhin et al., 2020; Thakur et al., 2020; Chang et al., 2020). Another common use of deep LM is cross-encoder, pass compared text pair directly in and use attention overall tokens to do prediction. In contrast to bi-encoder, cross encoder trains easier and is effective in low data for similarity and ranking tasks (Devlin et al., 2019; Yang et al., 2019).

Based on the same LM, however, bi-encoder and cross encoder have similar language understanding capabilities. To explain the difficulty in training bi-encoder not seen in cross-encoder, we look into the internal structure of pre-trained LM. We find LM like BERT directly out of pre-training has a non-optimal attention structure. In particular, they were not trained to aggregate sophisticated information into a single dense representation. We term effort during fine-tuning to adjust the LM internal activation to channel its knowledge out for the target task, *structural readiness*. We argue bi-encoder fine-tuning is inefficient due to the lacking structural readiness. Many updates are used to adjust model attention structure than learn good representation.

Based on our observations, we propose to address structural readiness during pre-training. We introduce a novel Transformer pre-training architecture, Condenser, which establishes structural readiness by doing LM pre-training actively CONdition on DENSE Representation. Unlike previous works that pre-train towards a particular task, Condenser pre-trains towards the bi-encoder structure. Our results show the importance of structural readiness. We experiment with sentence similarity tasks, and retrieval for question answering and web search. We find under low data setups, with identical test time architecture, Condenser yields sizable improvement over standard LM and shows

[1]Code available at https://github.com/luyug/Condenser

comparable performance to strong task-specific pre-trained models. With large training data, we find Condenser retriever optimize more easily, outperforming previous models trained with complicated techniques with a single round of negative mining.

2 Related Work

Transformer Bi-encoder LM pre-training followed by task fine-tuning has become one important paradigm in NLP (Howard and Ruder, 2018). SOTA models adopt the Transformer architecture (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Lan et al., 2020). One challenge for applying deep Transformer is their computation cost when used to retrieve text from large collections. Motivated by this, Reimers and Gurevych (2019) propose SBERT which trains bi-encoder from BERT and uses vector product for efficient sentence similarity comparison. Transformer bi-encoders were soon also adopted as dense retriever (Lee et al., 2019; Chang et al., 2020; Karpukhin et al., 2020; Gao et al., 2021b).

Dense Retrieval Dense retrieval compares encoded query vectors with corpus document vectors using inner product. While there are works on efficient cross-encoder (Gao et al., 2020; MacAvaney et al., 2020), such models are still too costly for full corpus retrieval. By pre-encoding the corpus into MIPS (Johnson et al., 2017; Guo et al., 2020) index, retrieval can run online with millisecond-level latency. An alternative is the recently proposed contextualized sparse retrieval model (Gao et al., 2021a). In comparison, dense retrieval is easier to use and backed by more matured software like FAISS (Johnson et al., 2017).

Pre-train Bi-encoder Lee et al. (2019) are among the first to show the effectiveness of Transformer bi-encoder for dense retrieval. They proposed to further pre-train BERT with Inverse Cloze Task (ICT). ICT uses pair of passage segment and full passage as pseudo training pair. Chang et al. (2020) find ICT and other related tasks are "key ingredients" for strong bi-encoders. Their results also show that models without pre-training fail to produce useful retrieval results under low data setups. Guu et al. (2020) propose to pre-train retriever and reader together for end-to-end QA system. The aforementioned methods are specialized *task specific* solutions for improving bi-encoder training based on contrastive loss. This paper provides an *explanation* for the learning issue and presents an architecture that establishes a universal solution using general language model pre-training. We also note that language model and contrastive pre-training are orthogonal ideas. In a follow-up work, we show further improved performance adding contrastive learning to Condenser language model pre-training (Gao and Callan, 2021).

Effective Dense Retriever Karpukhin et al. (2020) found carefully fine-tuning BERT can produce better results than earlier pre-trained dense retrieval systems. To further improve the end performance of dense retrievers, later works look into better fine-tuning techniques. Using a learned retriever to mine hard negatives and re-train another retriever with them was found helpful (Karpukhin et al., 2020; Qu et al., 2020). ANCE (Xiong et al., 2021) actively mines hard negatives once after an interval during training to prevent diminishing gradients. It allocates extra resources to update and retrieve from the corpus retrieval index repetitively. (Gao et al., 2021b) proposed to jointly learn a pair of dense and sparse systems to mitigate the capacity issue with low dimension dense vectors. Beyond fine-tuning, using more sophisticated knowledge distillation loss to learn bi-encoders based on soft labels has also been found useful (Chen et al., 2020; Lin et al., 2020). They first learn a teacher model and use its predictions at training time to optimize the dense retriever. These works all aim at producing better gradient updates during training, while Condenser aims at better initializing the model. We will also show the combined improvement of Condenser and hard negatives in experiments. Another line of works question the capacity of single vector representation and propose to use multi-vector representation (Luan et al., 2020). Capacity defines the performance upper bound and is one other issue than training (optimization), i.e. how to reach the upper bound.

Sentence Representation We'd also like to make a distinction from works in universal sentence representation and encoder (Kiros et al., 2015; Conneau et al., 2017; Cer et al., 2018). They are *feature-based* methods rather than fine-tuning (Houlsby et al., 2019). In evaluation, they focus on using the learned embedding as universal features for a wide range of tasks (Conneau and Kiela, 2018). This paper considers task-specific fine-tuning of the *entire* model and focuses on the target task performance.

3 Method

This section discusses the motivation behind Condenser, its design, and its pre-training procedure.

3.1 Preliminaries

Transformer Encoder Many recent state-of-the-art deep LM adopts the architecture of Transformer encoder. It takes in a text sequence, embed it and pass it through a stack of L self-attentive Transformer blocks. Formally, given input text $x = [x_1, x_2, ...]$, we can write iteratively,

$$h^0 = \text{Embed}(x) \tag{1}$$
$$h^l = \text{Transformer}_l(h^{l-1}) \tag{2}$$

Intuitively, Transformer blocks refine each token's representation conditioning on all tokens in the sequence to effectively embed them.

Transformer LM Pre-training Many successful Transformer Encoder LMs such as BERT are trained with masked language model (MLM) task. MLM masks out a subset of input tokens and requires the model to predict them. For a masked out token x_i at position i, its corresponding final representation h_i^L is used to predict the actual x_i. Training uses a cross-entropy loss,

$$\mathcal{L}_{\text{mlm}} = \sum_{i \in \text{masked}} \text{CrossEntropy}(W h_i^L, x_i) \tag{3}$$

A special token, typically referred to as CLS is prepended and encoded with the rest of the text.

$$[h_{cls}^0; h^0] = \text{Embed}([\text{CLS}; x]) \tag{4}$$
$$[h_{cls}^l; h^l] = \text{TF}_l([h_{cls}^{l-1}; h^{l-1}]) \tag{5}$$

Some models train CLS explicitly during pre-training, notably BERT's next sentence prediction (NSP; Devlin et al. (2019)), while others implicitly (Yang et al., 2019; Liu et al., 2019).

3.2 Issues with Transformer Encoder

Recall in Transformers, all tokens, including the CLS, receive information of other tokens in the sequence only with attention. Attention patterns, therefore, define how effective CLS can aggregate information. To understand the attentive behaviors of CLS, we borrow analysis of BERT from Clark et al. (2019): 1) in most middle layers, the CLS token has similar attention patterns as other text tokens and is not attended by other tokens, 2) until the last layer, CLS has unique broad attention over the entire sequence to perform NSP task. In other words, the CLS token remains dormant in many middle layers and reactivates only in the last round of attention. We argue that an effective bi-encoder should actively aggregate information of different granularity from the entire sentence through all layers, and this structure in standard pre-trained LM is not immediately ready for fine-tuning. We will verify this claim with experiments in section 4 and with quantitative analysis of attention of BERT, ICT, and the proposed Condenser in section 5.

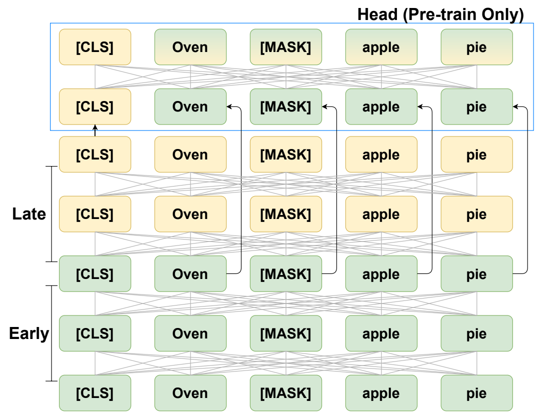


Figure 1: Condenser: We show 2 early and 2 late backbone layers here, in our experiments each have 6 layers. Condenser Head is dropped during fine-tuning.

3.3 Condenser

Building upon Transformer encoder LMs, which conditions on left and right context (Devlin et al., 2019), we present bi-encoder pre-training architecture Condenser, which CONdition actively on DENSE Representation in LM pre-training.

Model Design Like Transformer Encoder, Condenser is parametrized into a stack of Transformer blocks, shown in Figure 1. We divide them into three groups, L^e early encoder backbone layers, L^l late encoder backbone layers, and L^h Condenser head Layers. Inputs is first encoded by backbone,

$$[h_{cls}^{early}; h^{early}] = \text{Encoder}_{\text{early}}([h_{cls}^0; h^0]) \tag{6}$$
$$[h_{cls}^{late}; h^{late}] = \text{Encoder}_{\text{late}}([h_{cls}^{early}; h^{early}]) \tag{7}$$

Condenser Head The critical design is that we put a short circuit from early output to the head, which takes in a pair of *late-early* representations,

$$[h_{cls}^{cd}; h^{cd}] = \text{Condenser}_{\text{head}}([h_{cls}^{late}; h^{early}]) \tag{8}$$

We train with MLM loss with the head's output,

$$\mathcal{L}_{\text{mlm}} = \sum_{i \in \text{masked}} \text{CrossEntropy}(W h_i^{cd}, x_i) \tag{9}$$

We follow the masking scheme in Devlin et al. (2019) to combat train test difference.

Within Condenser, the late encoder backbone can further refine the token representations but can only pass new information through h_{cls}^{late}, the late CLS. The late CLS representation is therefore required to aggregate newly generated information later in the backbone, and the head can then *condition* on late CLS to make LM predictions. Meanwhile, skip connecting the early layers, we remove the burden of encoding local information and the syntactic structure of input text, focusing CLS on the global meaning of the input text. Layer numbers L^e and L^l control this separation of information.

Architecture of Condenser is inspired by Funnel Transformer (Dai et al., 2020), which itself is inspired by U-net (Ronneberger et al., 2015) from computer vision. Funnel Transformer reduces sequence length by a factor of 4 during forward and uses a 2-layer Transformer to decode the length compressed sequence onto a skip-connected full-length representation. Funnel Transformer was designed to speed up pre-training while our Condenser learns dense information aggregation.

Fine-tuning The Condenser head is a pre-train time component and is dropped during fine-tuning. Fine-tuning trains the late CLS h_{cls}^{late} and backpropagate gradient into the backbone. In other words, a Condenser reduces to its encoder backbone, or effectively becomes a Transformer encoder for fine-tuning; the head is only used to guide pre-training. During fine-tuning, Condenser has an identical capacity as a similarly structured Transformer. In practice, Condenser can be a drop-in weight replacement for a typical Transformer LM like BERT.

3.4 Condenser from Transformer Encoder

In this paper, we opted to initialize Condenser with pre-trained Transformer LM weight. This accommodates our compute budget, avoiding the huge cost of pre-training from scratch. This also gives us a direct comparison to the original LM. Given a pre-trained LM, we initialize the entire Condenser backbone with its weights and randomly initialize the head. To prevent gradient back propagated from the random head from corrupting backbone weights, we place a semantic constraint by performing MLM also with backbone late outputs,

$$\mathcal{L}_{\text{mlm}}^{c} = \sum_{i \in \text{masked}} \text{CrossEntropy}(Wh_i^{late}, x_i) \quad (10)$$

The intuition behind this constraint is that encoding per-token representations h^{late} and sequence representation h_{cls}^{late} share similar mechanism and will not interfere with each other. As a result, h^{late} can still be used for LM prediction. The full loss is then defined as a sum of two MLM losses,

$$\mathcal{L} = \mathcal{L}_{\text{mlm}} + \mathcal{L}_{\text{mlm}}^{c} \quad (11)$$

The output projection matrix W is shared between the two MLM losses to reduces the total number of parameters and memory usage.

4 Experiments

In this section, we first describe details on how to pre-train Condenser from BERT. Our fine-tuning experiments then look into the impacts of Condenser under low and high data setup. To evaluate low data, we sample smaller training sets similar to Chang et al. (2020), by sub-sampling the original train set. We keep dev/test sets unchanged across runs for direct comparison. We first validate our model with short sentence level tasks, then evaluate retrieval in open question answering and web search tasks following prior works (Chang et al., 2020; Xiong et al., 2021). We will examine how swapping original BERT with Condenser improves performance, and how the improvements compare to various improved training techniques.

4.1 Pre-training

We initialize Condenser backbone layers from the popular 12-layer BERT base and only a 2-layer head from scratch. Pre-training runs with procedures described in subsection 3.4. We use an equal split, 6 early layers, and 6 late layers. We pre-train over the same data as BERT: English Wikipedia and the BookCorpus. This makes sure BERT and Condenser differ only in architecture for direct comparison. We train for 8 epochs, with AdamW, learning rate of 1e-4 and a linear schedule with warmup ratio 0.1. Due to compute budget limit, we were not able to tune the optimal layer split, head size or train hyperparameters, but leave that to future work. We train on 4 RTX 2080ti with gradient accumulation. The procedure takes roughly a week to finish. After pre-training, we discard the Condenser head, resulting in a Transformer model of the *same* architecture as BERT. *All* fine-tuning experiments share this *single* pre-trained weight.

4.2 Sentence Similarity

Dataset We use two supervised data sets: Semantic Textual Similarity Benchmark(STS-b; Cer et al. (2017)) and Wikipedia Section Distinction (Ein Dor et al., 2018) adopted in Reimers and Gurevych (2019). The former is a standard sentence similarity task from GLUE (Wang et al., 2018) with a small training set (∼6K). The latter is large(∼1.8M) and has an interesting objective, to determine if a pair of sentences are from the same Wikipedia section, very similar to the BERT NSP task. Lan et al. (2020) argue NSP learns exactly topical consistency on the training corpus, i.e. Wikipedia. In other words, NSP is a close pre-training, if not training, task for Wiki Section Distinction. We report test set Spearman correlation for STS-b and accuracy for Wiki Section Distinction.

Compared Systems We compare with standard BERT and on STS-b, with BERT pre-trained with multiple NLI data sets with a popular carefully crafted 3-way loss (Conneau et al., 2017) from Reimers and Gurevych (2019)[2]. Non-BERT baselines are also borrowed from it.

Implementation We use the sentence transformer software and train STS-b with MSE regression loss and Wiki Section with triplet loss (Reimers and Gurevych, 2019). The training follows the authors' hyper-parameter settings.

Results Table 1 shows performance on STS-b with various train sizes. NLI pre-trained BERT and Condenser consistently outperform BERT and has a much larger margin with smaller train sizes. Also, with only 500 training pairs, they outperform the best Universal Sentence Encoder(USE) baseline.

For Wiki Section, in Table 2 we observe almost identical results among BERT and Condenser models, which outperform pre-BERT baselines. Meanwhile, even when training size is as small as 1K, we observe only about 10% accuracy drop than training with all data. Without training with the NSP task, Condenser remains effective.

4.3 Retrieval for Open QA

In this section, we test bi-encoders with open QA passage retrieval experiments (Chang et al., 2020; Karpukhin et al., 2020). Compared to the sentence level task, search tasks explicitly use the learned structure of the embedding space, where similarity corresponds to the relevance between a pair of query, passage. We adopt the DPR (Karpukhin et al., 2020) setup, fine-tune LM with a contrastive loss in training, computing for query q, the negative log likelihood of a positive document d^+ against a set of negatives $\{d_1^-, d_2^-, ..d_l^- ..\}$.

$$\mathcal{L} = -\log \frac{\exp(s(q, d^+))}{\exp(s(q, d^+)) + \sum_l \exp(s(q, d_l^-))} \tag{12}$$

Negatives can come from various sources: random, top BM25, hard negatives, or sophisticatedly sampled like ANCE. We conduct low data experiments with BM25 negatives to save compute and use mined hard negatives (HN) in full train experiments.

Dataset We use two query sets, Natural Question(NQ; Kwiatkowski et al. (2019)) and Trivia QA(TQA; Joshi et al. (2017)), as well as the Wikipedia corpus cleaned up and released with DPR. NQ contains questions from Google search and TQA contains a set of trivia questions. Both NQ and TQA have about 60K training data post-processing. We refer readers to Karpukhin et al. (2020) for details. We adopt DPR evaluation met-

[2]These models are referred to as SBERT in the original paper. We use BERT for consistency with later discussions.

STS-b			
Model	**Spearman**		
GloVe	58.0		
Infersent	68.0		
USE	74.9		
Train Size	500	1K	FULL
BERT	68.6	71.4	82.5
BERT + NLI	76.4	76.8	84.7
Condenser	**76.6**	**77.8**	**85.6**

Table 1: **STS-b**: Spearman correlation on Test Set.

Wikipedia Section Distinction			
Model	**Accuracy**		
skip-thoughts	0.62		
Train Size	1K	10K	FULL
BiLSTM	n.a.	n.a.	0.74
BERT	0.72	0.75	0.80
Condenser	0.73	0.76	0.80

Table 2: **Wiki Section**: Accuracy on Test Set.

Model	Natural Question Top-20			Top-100			Trivia QA Top-20			Top-100		
BM25	59.1			73.7			66.9			76.7		
Train Size	1K	10K	FULL	1K	10K	FULL	1K	10K	FULL	1K	10K	FULL
BERT	66.6	75.9	78.4	79.4	84.6	85.4	68.0	75.0	79.3	78.7	82.3	84.9
ICT	**72.9**	**78.4**	**80.9**	**83.7**	**85.9**	**87.4**	73.4	77.9	79.7	**82.3**	84.8	85.3
Condenser	72.7	**78.3**	80.1	82.5	**85.8**	86.8	**74.3**	**78.9**	**81.0**	**82.2**	**85.2**	**86.1**

Table 3: Low data: Results on Natual Question and Triavia QA measured by Top-20/100 Hits. Models in this table are all trained with BM25 negatives. Results within 0.1 difference with the best are marked bold.

rics, report test set hit accuracy of Top-20/100.

Compared Systems For low data experiments, we compare BERT, ICT, and Condenser. We attempted to train ICT on our hardware for direct comparison but found the end result bad, due to the small batch size. We instead use ICT released by Lee et al. (2019) trained with 4096 batch size from BERT for more informative comparison.[3] For full train, we compare with lexical systems BM25 and GAR (Mao et al., 2020) and dense systems DPR (BERT), DPR with HN and ANCE. GAR uses a learned deep LM BART (Lewis et al., 2020) to expand queries. ANCE uses asynchronous corpus index update (Guu et al., 2020) to do multiple rounds of hard negative mining during training. We also compare with RocketQA (Qu et al., 2020), which is trained with an optimized fine-tuning pipeline that combines hard negative, large (1024) batch, supervision from cross-encoder, and external data.

Implementation We train Condenser systems using the DPR hyper-parameter setting. We use a single RTX 2080ti and employ the gradient cache technique (Gao et al., 2021c) implemented in the GC-DPR toolkit[4] to perform large batch training with the GPU's limited memory. As DPR only released Natural Question hard negatives, we use theirs on Natural Question and mine our own with a Condenser retriever on TriviaQA.

Results In Table 3, we record test set performance for NQ and TQA with low data. We observe ICT and Condenser both outperform vanilla BERT, by an especially large margin at 1K training size, dropping less than 10% compared to full-size training for Top-20 Hit and less than 5% for Top-100. The improvement is more significant when considering the gain over unsupervised BM25. ICT and Condenser show comparable performance, with ICT slightly better on NQ and Condenser on TQA. This also agrees with results from Lee et al. (2019), that ICT specializes in NQ. The results suggest general LM-trained Condenser can be an effective alternative to task-specific pre-trained model ICT.

In Table 4, we compare Condenser trained with full training data with other systems. On NQ, dense retrievers all yield better performance than lexical retrievers, especially those that use hard negatives. We see Condenser performs the best for Top-20 and is within 0.1 to RocketQA for Top-100, without requiring the sophisticated and costly training pipeline. On TQA, we see GAR, lexical with deep LM query expansion, perform better than all dense systems other than Condenser. This suggests TQA may require granular term-level signals hard to capture for dense retrievers. Nevertheless, we find Condenser can still capture these signals and perform better than all other lexical and dense systems.

Model	NQ Top-20/100		TQA Top-20/100	
BM25	59.1	73.7	66.9	76.7
GAR	74.4	85.3	80.4	85.7
DPR	78.4	85.4	79.3	84.9
DPR + HN	81.3	87.3	80.7	85.8
ANCE	81.9	87.5	80.3	85.3
RocketQA	82.7	**88.5**	n.a.	n.a.
Condenser	**83.2**	**88.4**	**81.9**	**86.2**

Table 4: Full train for Natural Question and Trivia QA. Results not available are denoted 'n.a.' Results within 0.1 difference with the best are marked bold.

4.4 Retrieval for Web Search

In this section, we examine how Condenser retriever performs on web search tasks. The setup is similar to open QA. One issue with web search data sets is that they are noisier, containing a large number of false negatives (Qu et al., 2020). We investigate if Condenser can help resist such noise.

[3]A detailed discussion of this choice of ICT is in A.3

[4]`https://github.com/luyug/GC-DPR`

Model	MS-MARCO Dev MRR@10			MS-MARCO Dev Recall@1000			DL2019 NDCG@10		
BM25	0.184			0.853			0.506		
Train Size	1K	10K	FULL	1K	10K	FULL	1K	10K	FULL
BERT	0.156	0.228	0.309	0.786	0.878	0.938	0.424	0.555	0.612
ICT	0.175	0.251	0.307	0.847	0.905	0.945	0.519	0.585	0.624
Condenser	**0.192**	**0.258**	**0.338**	**0.852**	**0.914**	**0.961**	**0.530**	**0.591**	**0.648**

Table 5: Low data: Performacne is measured by MRR@10, Recall@1k. Models in this table are all trained with BM25 negatives.

As passage retrieval is the focus of the paper, we defer discussion of long document retrieval to A.4.

Dataset We use the MS-MARCO passage ranking dataset (Bajaj et al., 2018), which is constructed from Bing's search query logs and web documents retrieved by Bing. The training set has about 0.5M queries. We use corpus pre-processed and released with RocketQA. We evaluate on two query sets: MS-MARCO Dev[5] and TREC DL2019 queries. We report on Dev official metrics MRR@10 and Recall@1k, and report on DL2019 NDCG@10.

Implementation We train with the contrastive loss with a learning rate of 5e-6 for 3 epochs on a RTX2080ti. We pair each query with 8 passages as Luan et al. (2020) and use a total batch of 64 passages. Low data experiments use BM25 negatives and full data experiments use hard negatives mined with BM25 negative trained Condenser.

Compared Systems For low data settings, we again compare BERT, ICT, and Condenser. Here, all the three are not trained on the MS-MARCO corpus; we examine their generalization capability. For full training setup, we compare with lexical system BM25, deep LM augmented lexical systems DeepCT (Dai and Callan, 2019) and DocT5Qry (Nogueira and Lin, 2019), and dense systems, ANCE, TCT (Lin et al., 2020) and ME-BERT (Luan et al., 2020). TCT also aims at improving training like ANCE, but by replacing contrastive loss fine-tuning with knowledge distillation. ME-BERT uses BERT large variant as encoder, three times larger than LMs used in other systems, and represents passage with multiple vectors. It gets higher encoder and embedding capacity but has higher costs in train, inference, and retrieval. Since the full RocketQA system uses data external to MS-MARCO, for a fair comparison, we include the variant without external data in the main result Table 6 and separately compare Condenser with all RocketQA variants in Table 7.

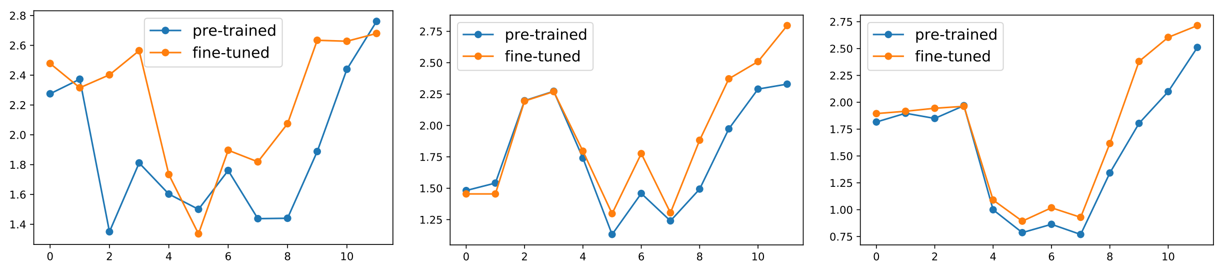
[5]The test set was hidden; MS-MARCO organizers discourage multi submissions but recommend studies over Dev set.

Model	MS-MARCO Dev MRR@10	MS-MARCO Dev R@1K	DL2019 NDCG@10
BM25	0.189	0.853	0.506
DeepCT	0.243	0.909	0.572
DocT5Qry	0.278	0.945	0.642
BERT	0.309	0.938	0.612
BERT + HN	0.334	0.955	0.656
ME-BERT	0.334	n.a.	0.687
ANCE	0.330	0.959	0.648
TCT	0.335	0.964	0.670
RocketQA*	0.364	n.a.	n.a.
Condenser	**0.366**	**0.974**	**0.698**

Table 6: Full train setup on MS-MARCO. Results not available are denoted 'n.a.' *: RocketQA variant here is not trained with external data.

Results In Table 5, we again find in low data, ICT and Condenser initialized retriever outperforms BERT by big margins. As it gets to 10K training data, 2% of the full training set, all dense retrievers outperform BM25, with ICT and Condenser retaining their margin over BERT. Condenser can already show comparable performance in recall and NDCG to BERT trained on the full training set. We also observe that Condenser can outperform ICT at various train size, suggesting that the general LM pre-training of Condenser help it better generalize across domains than task-specific ICT.

In Table 6, we compare full train performance of various system. We see various training techniques help significantly improve over vanilla fine-tuning. Condenser can further outperform these models by big margins, showing the benefits brought by pre-training. Without involving complex training techniques, or making model/retrieval heavy, Condenser can already show slightly better performance than RocketQA.



(a) BERT (b) ICT (c) Condenser

Figure 2: Attention patterns in pre-trained v.s. fine-tuned BERT, ICT and Condenser.

	batch size	MRR@10
RocketQA		
Cross-batch	8192	0.333
+ Hard negatives	4096	0.260
+ Denoise	4096	0.364
+ Data augmentation	4096	**0.370**
Condenser		
w/o hard negatives	64	0.338
w/ hard negatives	64	0.366

Table 7: Comparison with RocketQA MARCO Dev.

We further give a comparison with RocketQA variants in Table 7 to understand more costly strategies: very large batch, denoise hard negatives, and data augmentation. RocketQA authors find mined hard negatives contain false negatives detrimental to bi-encoder training as shown in the table and propose to use cross-encoder to relabel and denoise them, a process however thousands of times more costly than hard negative mining. They further employ a data augmentation technique, using a cross encoder to label external data. Here, we see Condenser trained with batch size 64 and BM25 negatives has better performance than RocketQA with 8192 batch size. More importantly, Condenser *is able* to resist noise in mined hard negatives, getting a decent boost training with mined hard negatives, unlike RocketQA whose performance drops a lot without denoise. We see that Condenser removes the need for many sophisticated training techniques: it is only outperformed by the RocketQA variant that uses external data (data augmentation).

Interestingly, our runs of BERT (DPR) + HN have decent performance improvement over BERT in all retrieval tasks, sometimes better than active mining ANCE on both QA and Web Search. This contradicts the finding in RocketQA that directly mined hard negatives hurts performance. Recall our hard negatives are mined by Condenser retriever, which we conjecture has produced higher quality hard negatives. The finding suggests that mined hard negatives may not be retriever-dependent. There exist universally better ones, which can be found with a more effective retriever.

5 Attention Analysis

Condenser is built upon the idea that typical pre-trained LM lacks proper attention structure. We already see that we can fix the issue by pre-training with Condenser in the last section. In this section, we provide a more in-depth attention analysis: we compare attention behaviors among pre-trained/fine-tuned BERT, ICT, and Condenser. We use an analytical method proposed by Clark et al. (2019), characterizing the attention patterns of CLS by measuring its attention entropy. A higher entropy indicates broader attention and a lower more focused attention. Similar to Clark et al. (2019), we show CLS attention entropy at each layer, averaged over all heads, and averaged over 1k randomly picked Wikipedia sections.

In Figure 2, we plot attention from CLS of various models. We see in Figure 2a that BERT has a drastic change in attention pattern between pre-trained and fine-tuned models. This again confirmed our theory that typical Transformer Encoder LMs are not ready to be fined-tuned into bi-encoder, but need to go through big internal structural changes. In comparison, we see in Figures 2b, 2c that task-specific pre-trained ICT and LM pre-trained Condenser only have small changes, retaining general attention structure. In other words, ICT and Condenser both established structural readiness, but in very different ways. Both ICT and Condenser have broadening attention (increased entropy) in the later layers, potentially because the actual search task requires aggregating more high-level concepts than pre-training. The results here again confirm our theory, that a ready-to-use structure can be easier to train; their structures only need small changes to work as an effective bi-encoder.

6 Conclusion

Fine-tuning from a pre-trained LM initializer like BERT has become a very common practice in NLP. In this paper, we however question if models like BERT are the most proper initializer for bi-encoder. We find typical pre-trained LM does not have an internal attention structure ready for bi-encoder. They cannot effectively condense information into a single vector dense representation. We propose a new architecture, Condenser, which establishes readiness in structure with LM pre-training. We show Condenser is effective for a variety of tasks, sentence similarity, question answering retrieval, and web search retrieval. With low data, Condenser shows comparable performance to task-specific pre-trained models. It also provides a new pre-training perspective in learning effective retrievers than fine-tuning strategies. With sufficient training, Condenser and direct fine-tuning can be a lightweight alternative to many sophisticated training techniques.

Positive results with Condenser show that structural readiness is a fundamental property in easy-to-train bi-encoders. Our attention analysis reveals both Condenser and task-specific pre-trained model establish structural readiness, suggesting task-specific objective may not be necessary. Researchers can use this finding to guide the study of better LM for bi-encoder, for example, explore training Condenser with other LM objectives.

One big advantage of BERT is that after cumbersome pre-training for once, fine-tuning is easy with this universal model initializer. This is however not true for BERT bi-encoder, especially retriever, which needs careful and costly training. Condenser extends this benefit of BERT to bi-encoder. Practitioners on a limited budget can replace BERT with our pre-trained Condenser as the initializer to get an instant performance boost. Meanwhile, for those aiming at the best performance, training techniques and Condenser can be combined. As we have demonstrated the combined effect of hard negatives and Condenser, sophisticated but better techniques can be further incorporated to train Condenser.

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.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

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*.

Jiecao Chen, Liu Yang, Karthik Raman, Michael Bendersky, Jung-Jung Yeh, Yun Zhou, Marc Najork, Danyang Cai, and Ehsan Emadzadeh. 2020. DiPair: Fast and accurate distillation for trillion-scale text matching and pair modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2925–2937, Online. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. *ArXiv*, abs/1906.04341.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *ArXiv*, abs/1803.05449.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

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

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *ArXiv*, abs/2006.03236.

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.

Liat Ein Dor, Yosi Mass, Alon Halfon, Elad Venezian, Ilya Shnayderman, Ranit Aharonov, and Noam Slonim. 2018. Learning thematic similarity metric from article sections using triplet networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Melbourne, Australia. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval.

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, and Jamie Callan. 2021a. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042, Online. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021b. 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, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021c. 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.

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.

N. Houlsby, A. Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and S. Gelly. 2019. Parameter-efficient transfer learning for nlp. In *ICML*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

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.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer.

2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using tightly-coupled teachers. *ArXiv*, abs/2010.11386.

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.

Sean MacAvaney, F. Nardini, R. Perego, N. Tonellotto, Nazli Goharian, and O. Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

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

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

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.

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

Y. Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, X. Zhao, Daxiang Dong, Hua Wu, and H. Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *ArXiv*, abs/2010.08191.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2020. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

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*.

A Appendix

A.1 Hyper Parameters Settings

STS-b The training follows hyper-parameter settings in Reimers and Gurevych (2019), Adam optimizer, a learning rate of 2e-5 with linear schedule, and 4 epochs. For low data setup, we search best epoch number in {4,8} for BERT and apply those to all other pre-trained models.

Wikipedia Section Distinction The training follows hyper-parameter settings in Reimers and Gurevych (2019), Adam optimizer, a learning rate of 2e-5 with linear schedule and 1 epoch. For low data setup, we search best epoch number in {1,4,8} for BERT and apply those to all other pre-trained models.

Open QA We follow hyperparameter settings in Karpukhin et al. (2020), 128 batch size, 1 BM25 negative, in-batch negatives, 40 epochs, 1e-5 learning rate and linear schedule with warmup. Low data share the same setting as we found 40 epochs are enough for convergence.

Web Search We train with Adam optimizer, learning rate of 5e-6 for 3 epochs with a total batch size of 64: 8 query $\times$ 8 passages. For low data setup, we search best epoch number in {5, 10, 40} for BERT and apply those to all other pre-trained models.

A.2 Model Size

In our experiments, Condenser during fine-tuning has the same number of parameters as BERT base, about 100 M. Adding the head during pre-training, there are roughly 120 M parameters.

A.3 ICT Model

Our ICT model comes from Lee et al. (2019). It is trained with a batch size of 4096. ICT's effectiveness in low data setup was verified and thoroughly studied by Chang et al. (2020). Chang et al. (2020) also introduces two other pre-training tasks Body First Selection and Wiki Link Prediction. They heavily depend on Wikipedia like structure and knowledge of the structure during pre-training and therefore does not apply in general situations. Meanwhile, adding them improves over ICT by only around 1% and Chang et al. (2020) has not released their model checkpoints. Therefore we chose to use the ICT checkpoint.

Difficulties in reproducing these models come from the large batch requirement and the contrastive loss in ICT. Both Lee et al. (2019) and Chang et al. (2020) find it critical to use large batch: Lee et al. (2019) uses a 4096 batch and Chang et al. (2020) a 8192 batch. Both were trained with Google's cloud TPU. In comparison, our GPU can fit a batch of only 64. The contrastive loss uses the entire batch as the negative pool to learn the embedding space. Using gradient accumulation will reduce this pool size by several factors, leading to a bad pre-trained model. In comparison, our Condenser is based on instance-wise MLM loss and can naively use gradient accumulation.

We convert the original Tensorflow Checkpoint into Pytorch with huggingface conversion script. We don't use the linear projection layer that maps the 768 BERT embedding vector to 128 so that the embedding capacity is kept the same as retrievers in Karpukhin et al. (2020).

Model	MS-MARCO Dev MRR@100	DL2019 NDCG@10
BM25	0.230	0.519
DeepCT	0.320	0.544
BERT	0.340	0.546
ME-BERT	n.a.	0.588
ANCE	0.382	**0.615**
Condenser	0.375	0.569
Condenser + HN	**0.404**	0.597

Table 8: Full train setup on MS-MARCO Document. Results not available are denoted 'n.a.'

A.4 Document Retrieval

Recent works (Xiong et al., 2021; Luan et al., 2020) explored retrieving long documents with the MS-MARCO document ranking dataset (Bajaj et al., 2018). There are several issues with this data set. The training set is not directly constructed but synthesizing from the passage ranking data set label. Xiong et al. (2021) find that the judgment in its TREC DL2019 test set biased towards BM25 and other lexical retrieval systems than dense retrievers. Meanwhile, Luan et al. (2020) find single vector representation has a capacity issue in encoding long documents. To prevent these confounding from affecting our discussion, we opted to defer the experiment to this appendix. Here we use two query sets, MS-MARCO Document Dev and TREC DL2019. We report official metrics MRR@100 on Dev and NDCG@10 on DL2019. Results are recorded in Table 8. Condenser improves over BERT by a large margin and adding HN also boosts its performance. Condenser + HN performs the best on Dev. On the other hand, we see ANCE is the best on DL2019. We conjecture the reason is that use of BM25 negatives in many systems is not favorable towards DL2019 labels that favor lexical retrievers. The multi rounds of negative mining help ANCE get rid of the negative effect of BM25 negatives.

A.5 Engineering Detail

We implement Condenser (from BERT) in Pytorch (Paszke et al., 2019) based on the BERT implementation in huggingface transformers package (Wolf et al., 2019). As our adjustments go only into the model architecture and the LM objective is kept unchanged, we only need to modify the modeling file and reuse the pre-training pipeline from huggingface.

A.6 Link To Datasets

Sentence Similarity Cleaned up version can be found in the sentence transformer repo `https://github.com/UKPLab/sentence-transformers`.

Open QA We use cleaned up open qa data from DPR `https://github.com/facebookresearch/DPR/`.

Web Search MS-MARCO data can found on its homepage `https://microsoft.github.io/msmarco/`.