Is Your Language Model Ready for Dense Representation Fine-tuning?

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

Abstract

Pre-trained language models (LM) have become go-to text representation encoders. Prior research used deep LMs to encode text sequences such as sentences and passages into single dense vector representations. These dense representations have been used in efficient text comparison and embedding-based retrieval. However, dense encoders suffer in low resource situations. Many techniques have been developed to solve this problem. Despite their success, not much is known about why this happens. This paper shows that one cause lies in the readiness of the LM to expose its knowledge through dense representation in fine-tuning, which we term Optimization Readiness. To validate the theory, we present *Condenser*, a general pre-training architecture based on Transformer LMs, to improve dense optimization readiness. We show that fine-tuning from Condenser significantly improves performance for small and/or noisy training sets.[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 (Vaswani et al., 2017) LMs like BERT (Devlin et al., 2019) are the state-of-the-art (Liu et al., 2019; Yang et al., 2019; Lan et al., 2020). The latest dense encoders, or bi-encoders are fine-tuned from Transformer LMs to encode text into a single vector (Reimers and Gurevych, 2019; Karpukhin et al., 2020). 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 inference test efficiency, bi-encoders are not sample efficient, taking a big performance hit in low resource situations (Karpukhin et al., 2020; Thakur et al., 2020). In particular, Thakur et al. (2020) found that bi-encoders significantly underperform their self-attentive counterpart (cross-encoder) in low resource settings.

Solution for data scarcity, such as heuristic or semi-supervised data augmentation, can help boost bi-encoder performance under low resource. However, the exact cause for data inefficiency is unknown. To explain it, this paper presents a theory about LM readiness towards fine-tuning tasks. We describe two types of Readiness: 1) Knowledge Readiness, an LM's capability to understand the target task's language patterns, and 2) Optimization Readiness, required effort to adjust the LM to channel its knowledge out for the target task. Knowledge Readiness naturally comes from masked language model (MLM; Devlin et al. (2019)) training. It proves to be effective and critical for learning general language knowledge. However, given the disparity between MLM pre-trained Transformer and bi-encoder, we argue popular LM like BERT lacks bi-encoder Optimization Readiness.

In this paper, to test out our theory and also guided by it, we introduce a novel *general* Transformer Encoder pre-training architecture, Condenser, which boosts Optimization Readiness by performing MLM predictions actively CONdition on DENSE Representation. Our results show the importance of Optimization Readiness: we experiment extensively with sentence representation, question answering (QA) and web search tasks, and find with identical test time architecture, pre-training time task and data, Condenser converted from standard Transformer LM yields sizable improvement using identical fine-tuning setup without any data augmentation.

[1] Our pre-training code is at https://github.com/luyug/Condenser

Our contribution in this paper includes: 1) present a theory that can guide effective general LM pre-training for bi-encoder, 2) propose a novel implementation of the theory, Condenser, that excels under low resource setup, 3) demonstrate how to stably convert standard pre-trained Transformer LMs into Condenser with low computation cost.

2 Related Work

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 and MLM task (Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020). Fine-tuning LM into bi-encoder has also proved effective for various tasks (Reimers and Gurevych, 2019; Karpukhin et al., 2020).

The data inefficiency of bi-encoder has been a long-known issue. Many successful solutions have been proposed to boost the quality of bi-encoder under low resource. For sentence embedding, Reimers and Gurevych (2019) perform transfer learning from NLI. Thakur et al. (2020) uses semi-supervised training signal from bi-encoder. For retrieval, Lee et al. (2019) consider heuristic Inverse Cloze Task (ICT) which emulates search task. Chang et al. (2020) argues the use of ICT and other related tasks are "key ingredients" for strong bi-encoders. Guu et al. (2020) further consider building inductive bias into the LM by adding a retrieval component to LM pre-training. The aforementioned methods are *specialized solution* for low resource bi-encoder training. This paper provides an *explanation* for the issue and presents a general architecture to solve the problem at LM pre-training time. While this paper studies general pre-training architectures of encoder-only LM, we note modeling adjustment of deep LM for target task is not a new idea: when it comes to generation task, pre-trained encoder-decoder models have been shown effective(Lewis et al., 2020a; Raffel et al., 2020).

We'd also like to make a distinction from works in universal sentence representation (Kiros et al., 2015; Conneau et al., 2017; Cer et al., 2018). In evaluation, they focus on using the learned embedding as universal features that are linearly meaningful 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. Fine-tuning trains the entire deep non-linear model and is found more effective for specific end task performance (Peters et al., 2019).

3 Fine-tuning Readiness

In this section, we provide a conceptual argument for what an LM learns during pre-training and during fine-tuning. We discuss how the (in)sufficiency of the former to the latter can affect an LM's readiness for (dense) fine-tuning. We will present quantitative analysis of this theory in section 5.

LM pre-training is based on distributed hypothesis. By unsupervised training on massive text, the LM builds up general language understanding *knowledge* on the training corpus. The implementation of the LM, on the other hand, defines the LM's *structure*, output and internal behavior. For example, bi-directional Transformer LM output position-wise contextualized word embedding with permutation invariant self attention operations. In BERT, the CLS token carries information about sentence pair relations.

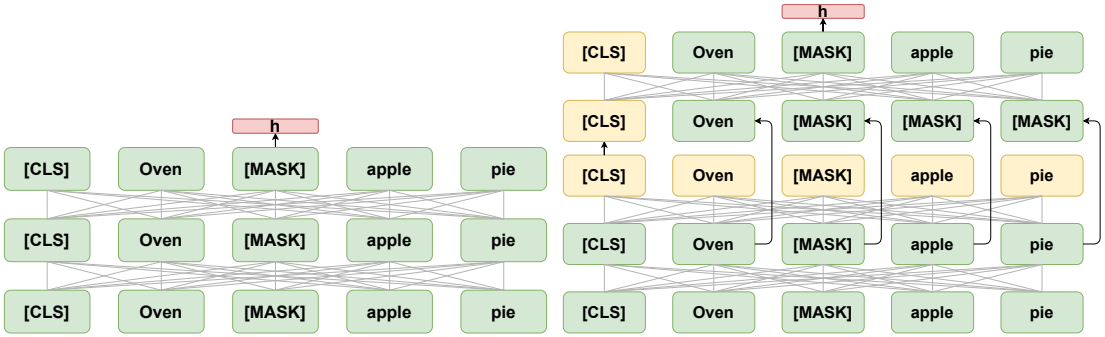
During fine-tuning, the supervised training data brings in task-specific language knowledge and task semantic (what to predict). Meanwhile, it also defines the task structure, guiding a pre-trained LM to optimize its internal computation to channel out its knowledge and produce effective output. The differences in general and task knowledge, and in LM and task structures, define two types of readiness,

- Knowledge Readiness
- Optimization Readiness

In this paper, we focus on general pre-training and do not consider task semantic in readiness but assume it will be learned only in fine-tuning.

Knowledge Readiness is very well-studied: BERT demonstrates substantial gain on general language understanding evaluation (GLUE; Wang et al. (2018)) with LM pre-training, while post-BERT models typically see improvements from expanded training corpus (Liu et al., 2019). Optimization Readiness is less problematic for downstream tasks such as token level or sentence pair classification, as the original LM training is sufficient. Optimization Readiness however becomes a significant issue for bi-encoders as the process of *general*[2] *text aggregation* into a single vector has

[2]Beyond relational information as in next sentence prediction



(a) Transformer Encoder LM (b) Condenser

Figure 1: Illustration of Transformer Encoder LM and Condenser LM.

never been baked into the LM. We hypothesize the Optimization Readiness issue causes difficulties for bi-encoder in low resource setups. We explicitly address Optimization Readiness in subsection 4.2 with a novel architecture Condenser. Condenser rewires a Transformer LM to learn information aggregation into the CLS vector.

4 Methodologies

4.1 Preliminaries

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

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

Intuitively, each Transformer blocks refines each token's representation conditioning on all tokens in the sequence and the output becomes a effective contextualized representation (Peters et al., 2018).

Transformer LM Pre-training Many successful Transformer LM such as BERT (Devlin et al., 2019) is trained with MLM. MLM masks out a subset of input token and require the model to predict them. Formally, 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 representation explicitly during pre-training, notably BERT using next sentence prediction (Devlin et al., 2019; Lan et al., 2020), while others only at fine-tuning time (Yang et al., 2019; Liu et al., 2019).

Bi-encoder A bi-encoder based on pre-trained LM typically uses linear transformed CLS representation and fine-tune it on some downstream task. Formally, given matrix $A^{e \times d}$, we get $v \in \mathbb{R}^e$

$$v = A h_{cls}^L \tag{6}$$

A pair of such models, bi-encoder, encode a pair of texts into two vectors for similarity comparison.

4.2 Condenser

Our readiness theory calls for a model capable of condensing text sequence information in a single vector, being learned at pre-training time. In this section, we show how to achieve this by rewiring a feed-forward Transformer LM so that MLM is performed by CONditioning on the DENSE Representation (Condenser).

Model design We design Condenser to closely resemble the popular Transformer encoder LM like BERT to help provide more controlled understanding. One critical difference in Condenser is that it places the CLS token at the center of MLM prediction by actively conditioning on it. Condenser is parametrized into a stack of Transformer encoder 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. We first run the input through the backbone,

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

$$[h_{cls}^{late}; h^{late}] = \text{Encoder}_{\text{late}}([h_{cls}^{early}; h^{early}]) \quad (8)$$

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

$$[h_{cls}^{cd}; h^{cd}] = \text{Condenser}_{\text{head}}([h_{cls}^{late}; h^{early}]) \quad (9)$$

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

$$\mathcal{L}_{\text{mlm}} = \sum_{i \in \text{masked}} \text{CrossEntropy}(W h_i^{cd}, x_i) \quad (10)$$

Here in the Condenser, the late encoder backbone can further refine the token representations but can only pass those information through h_{cls}^{late}, the late CLS representation. The CLS representation is therefore required to aggregate newly generated information later in the backbone and redistribute it to tokens in the Condenser head. As Transformer LMs process local information in earlier layers and global in the later layers (Clark et al., 2019), the Condenser CLS will be focused on the global information generated in later backbone. Layer numbers L^e and L^l control what CLS learns and should be determined by experiments. For familiar readers, we'd like to point out that Condenser Head draws inspiration from Funnel Transformer's decoder (Dai et al., 2020) with the major difference that their decoder is used to inflate length-compressed sequence. The head optimization readies the model: to take advantages of the latter layers, it is forced to aggregate information into the CLS. Meanwhile, training on MLM task, Condenser remains general pre-training.

Fine-tuning The Condenser head is a pre-train time architecture and is dropped during fine-tuning which trains h_{cls}^{late} and back propagates gradient only into the backbone. In other words, a Condenser reduces to its backbone, or effectively becomes a Transformer for fine-tuning. Formally during fine-tuning, it has the identical functional space as a same structured Transformer. In practice, it can be used as a drop-in weight replacement for a typical Transformer LM.

4.3 Condenser from Transformer

In this paper, we opt to initialize Condenser with pre-trained Transformer LM weight. This accommodates our compute budget, avoiding the huge cost of pre-training from scratch. Despite heavy parameter tuning cost of new Transformer architectures (Nguyen and Salazar, 2019), in section 5 we empirically find this approach allow us to train stably with BERT's hyper-parameters. Meanwhile, initialization from pre-trained weight provides a better alignment of knowledge between compared Condenser and Transformer. 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 with backbone late outputs,

$$\mathcal{L}_{\text{mlm}}^c = \sum_{i \in \text{masked}} \text{CrossEntropy}(W h_i^{late}, x_i) \quad (11)$$

The full loss is defined as a sum of two MLM losses,

$$\mathcal{L} = \mathcal{L}_{\text{mlm}} + \mathcal{L}_{\text{mlm}}^c \quad (12)$$

5 Experiments

In this section, we first describe the details in Condenser pre-training starting from a BERT LM. Our fine-tuning experiments then look into the impact of conversion from BERT LM into Condenser BERT. We first look into sentence-level tasks and turn to passage-level question answering and web search tasks. For each experiment, we fine-tune on different size-reduced training sets as well as full set to study the effects of Optimization Readiness. While the major comparison is between BERT and Condenser to validate our readiness theory, we also include popular non-dense baselines as references.

All of our Condenser fine-tuning runs follow the philosophy of *direct drop-in replacement*: we fine-tune Condenser BERT with identical setup, e.g. hyper parameters, optimized for BERT training, by either inheriting setups from public software (in sentence, open QA), or tuning the hyper parameter for BERT (in web search). The goal is to demonstrate Condenser robustness and versatility in existing pipelines and therefore the importance of Optimization Readiness fostered in pre-training.

5.1 Pre-training

We initialize Condenser backbone layers from BERT 12-layer base model and only 2-layer head

from scratch. Pre-training runs with procedures described in subsection 4.3, denoted as BERT + CD. 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 Knowledge Readiness is kept unchanged in Condenser. We train for 8 epochs, with Adam, learning rate of 1e-4 and a linear schedule with warmup ratio 0.1. We were not able to tune the optimal layer split or train hyper parameters due to compute budget limit, but leave that to future work. We train on 4 RTX 2080ti with gradient accumulation.

After pre-training, we discard the Condenser head [3], resulting in a Transformer model of the *same* architecture as BERT.

5.2 Sentence Representation

We start off testing some sentence-level tasks where an LM is used to encode a short text sequence into a single vector. The vector similarity then corresponds to the strength of some relation between two sentences.

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) and also borrow non-BERT baselines from it. The former is a standard sentence similarity task from GLUE with a relatively 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 BERT NSP task. Lan et al. (2020) argue NSP learns exactly topical consistency on Wikipedia. Given the similarity between pre-training and target task, we expect the original BERT to be Optimization Ready for Wiki Section. We report test set Spearman correlation for STS-b and accuracy for Wiki Section.

Implementation We use the sentence transformer software and train STS-b with mean-squared-error regression loss and Wiki Section with triplet Hinge loss (Reimers and Gurevych, 2019). The CLS token is used as dense embedding. The training follows the authors' released hyper-parameter settings, Adam optimizer, a learning rate of 2e-5 with linear schedule, 4 epochs for STS-b and 1 epoch for Wiki section. For Wiki Section, we train 4 epochs for reduced training size.

Results Table 1 shows performance on STS-b with various train sizes. With Condenser BERT consistently outperforms original BERT and has a much larger margin with smaller train sizes. As in Figure 2, the margin closes with more training data but remains non-trivial with full training set. Also, with as little as 500 training pairs, Condenser BERT outperforms the SNLI supervised Universal Sentence Encoder(USE) baseline.

For Wiki Section, as expected, in Table 2 we observe almost identical result between Condenser BERT and BERT, both of 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. The results confirm our theory that when an LM is both knowledge and optimization ready, fine-tuning becomes much easier.

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 + CD	**76.6**	**77.8**	**85.6**

Table 1: **STS-b**: Results are measures by 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
BERT + CD	0.73	0.76	0.80

Table 2: **Wikipedia Section Distinction** : Results are measures by Accuray on Test Set.

5.3 Open QA

In this section, we test bi-encoders as dense retrievers for open QA. 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

[3] An exception is the continue training scheme.

Open QA												
Model	**Natural Question**						**Trivia QA**					
	Top-20			Top-100			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
BERT + CD	**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: **Open QA**: Results on Natual Question and Triavia QA measured by Top-20/100 Hits over various training sizes. The BERT results correspond in practice to the vanilla DPR setup.

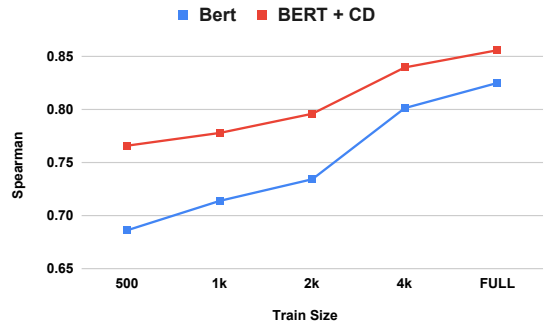


Figure 2: **STS-B**: Performance measured by Spearman correlation on various training size.

query, passage. Recent works like Dense Passage Retrieval(DPR; Karpukhin et al. (2020) adopt a contrastive loss in training, computing for query q, negative log likelihood of a positive document d^+ over a set of negative documents $\{d_1^-, d_2^-, ..d_l^-..\}$.

$$\mathcal{L} = -\log \frac{\exp(s(q, d^+))}{\exp(s(q, d^+)) + \sum_l \exp(s(q, d_l^-))} \tag{13}$$

For inference, all passages in a corpus are encoded into an index, queried with inner product search. The dense retriever can be integrated into different styles of open QA pipeline like DPR or RAG (Lewis et al., 2020b). Here, we focus on understanding retrieval quality.

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 processing details. We adopt DPR evaluation metrics, hit accuracy of Top-20 and Top-100.

Implementation We use DPR and train following published setups: 128 batch size, 1 BM25 negative, in-batch negatives, 40 epochs, 1e-5 learning rate and linear schedule with warmup. We train on a single RTX 2080ti using gradient check-pointing.

Results In Table 3, we record test set performance for NQ and TQA. In general, we observe similar patterns on both data sets: towards the smaller training side, Condenser BERT achieves a large performance advantage over BERT, 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. Trends on NQ are also plotted in Figure 3 where with increasing training size, the performance margin narrows, down finally to 2%, suggesting sufficient training can make up for missing Optimization Readiness and help learn the task structure.

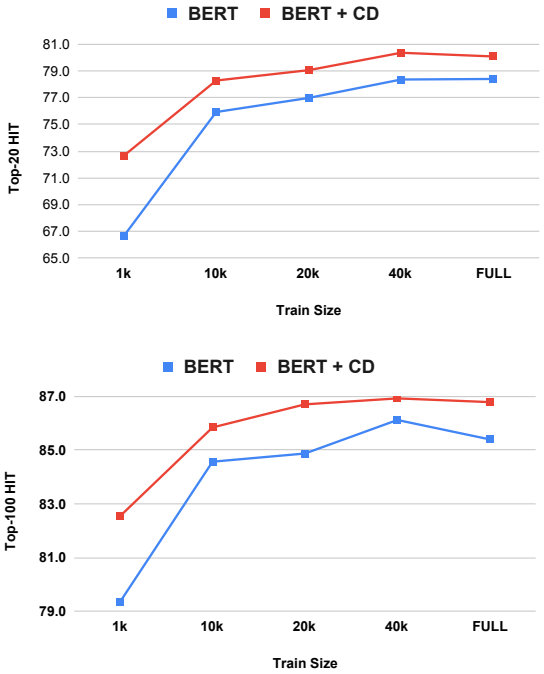


Figure 3: **NQ**: Effect of training size. Performance measured by Top-20 Hit (upper) and Top-100 hit (lower).

MS-MARCO Passage Ranking									
Model	MRR@10			Recall@100			Recall@1K		
BM25	0.184			0.657			0.853		
Train Size	1K	10K	FULL	1K	10K	FULL	1K	10K	FULL
DeepCT	n.a.	n.a.	0.243	n.a.	n.a.	n.a.	n.a.	n.a.	0.909
BERT	0.146	0.221	0.305	0.539	0.688	0.810	0.749	0.860	0.933
BERT + CD	0.175	0.251	0.320	0.621	0.737	0.829	0.820	0.898	0.944
BERT + CD + CT	**0.193**	**0.265**	**0.335**	0.653	**0.760**	**0.857**	0.842	**0.911**	**0.958**

Table 4: **Web Search - MS-MARCO**: Performacne is measured by MRR@10, Recall@100/1k. Results not available are denoted 'n.a.'. 'CD' refers to general Condenser pre-training and 'CT' to continue pre-training on the MS-MARCO corpus.

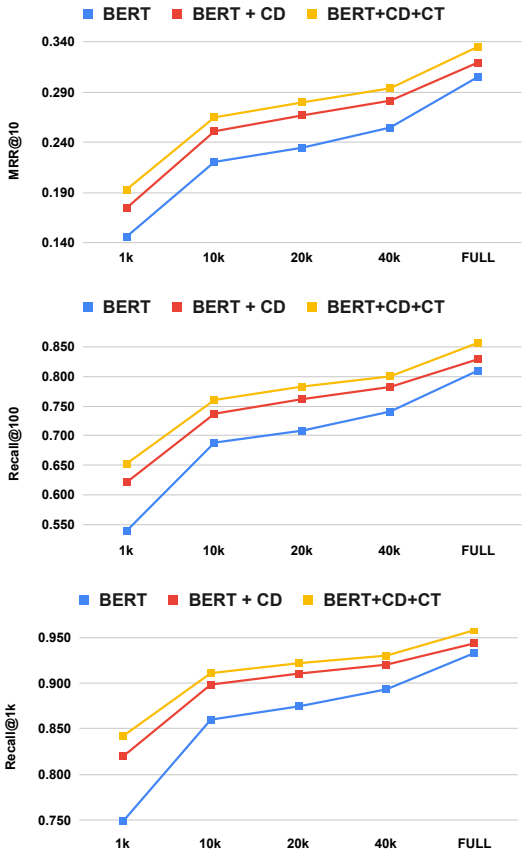


Figure 4: **MS-MARCO**: Effect of training size. 'CD' refers to general Condenser pre-training and 'CT' to continue pre-training on the MS-MARCO corpus.

5.4 Web Search

The model setup of web search is very similar to open QA. Some big differences lie in the data sets. Compared to Wikipedia in Open QA, web search corpus covers a wider variety of passages from the web. The data set judgments are typically generated from user click information. As a result, both training and testing data will be noisier, containing a sizable number of false negatives and potentially also false positives. In this section, we examine how various bi-encoders work under these noises. Meanwhile, the retrieval results are commonly presented to the user or to a reranker for reranking. Note that direct comparison between open QA and web search results is less meaningful.

Dataset We use the MS-MARCO passage ranking dataset (Bajaj et al., 2018), which has been used in evaluating many information retrieval systems. It is constructed from Bing's search query logs and web documents retrieved by Bing. The training set has about 0.5M queries. We include BM25 and BERT augmented term weight BM25, DeepCT (Dai and Callan, 2019) for performance reference. We report on the Dev set[4] MS-MARCO official metrics MRR@10 and Recall@1k, and in addition Recall@100 for limited reranking budgets.

Continue Training For MS-MARCO, there is also a Knowledge Readiness issue: pre-training corpora (Wikipedia and BookCorpus) are different from MS-MARCO web passages collection. We, therefore, add an additional pre-training step where we continue pre-training Condenser BERT on the passage collection. This procedure is *unsupervised* and happens *after* general pre-training and *before* fine-tuning. This experiment aims at demonstrating: 1) the Condenser architecture also inherits the continue training capability, 2) Knowledge Readiness is also critical.

Implementation We train using a contrastive loss with learning rate of 5e-6 with a linear schedule and a 0.1 warmup ratio on a single RTX2080[5]. We pair each query with 8 passages and use a total batch of 64 passages. For continue-training, we

[4]The test was hidden and MS-MARCO organizers recommend performing studies over the Dev set.

[5]Hyper parameters are tuned for BERT on this hardware.

inherit the pre-training setup but use a decreased learning rate of 5e-6.

Result In Table 4, we again find a very similar comparison in low resource end, Condenser BERT significantly outperforms BERT while adding continue training further improves performance. We however observe big drops in all bi-encoder models with small train sizes. We believe this is due to the fact that MS-MARCO training is not only much larger but also noisier than NQ or TQA. In Figure 4, we see across all training sizes, there is a clear separation among bi-encoders from BERT, Condenser BERT (BERT + CD), continue-trained Condenser (BERT + CD + CT): the differences show the importance of both Optimization and Knowledge Readiness. We also find that with full training, non-trivial performance differences still exist among the three bi-encoders towards the top of the ranking: when the training set is noisier, better initialization, knowledge and optimization ready, can give better performance even with large training data.

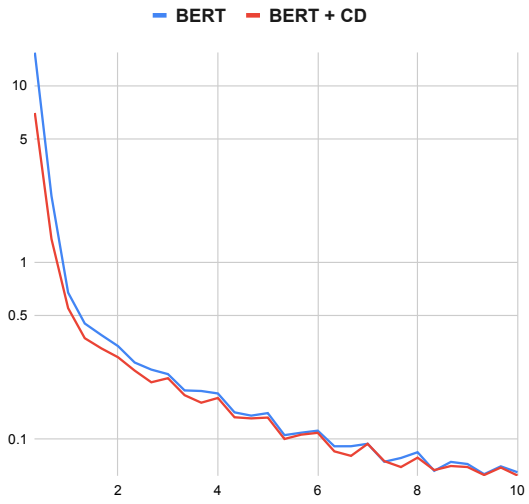


Figure 5: **NQ**: Loss in first 10 epochs log-scale.

5.5 Discussion

We want to be careful about drawing conclusions on what we learn from the previous experiments. With Condenser, we keep the same MLM loss and therefore the LM acquires a similar general language knowledge. Meanwhile, the Condenser architecture has an architecture that forces itself to build up dense representation during pre-training, and consequently optimization ready itself. In Figure 5, we plot training loss of BERT v.s. Condenser BERT. In early epochs, Condenser models consistently have lower loss than original BERT. In comparison, earlier approaches train over "pseudo" data to achieve a conversion from standard LM to dense representation encoder. Based on our results, we believe one of the fundamental effects of this fine-tuning is to learn the task structure and make up for missing Optimization Readiness. However, as the training objective is no longer language modeling, this fine-tuning also train a task semantic captured by the pseudo data. This explains why crafting proper pre-training data is crucial for model sucess (Chang et al., 2020; Thakur et al., 2020).

6 Conclusion

Fine-tuning, as the original word, refers to the process of making small adjustments to achieve better performance. With respect to deep LM, the adjustments are not necessarily small. In this paper, we found, when it comes to bi-encoder, standard LM spends big efforts changing internal behavior to fit task structure. We present a Readiness theory of optimization and knowledge, describing the required effort for fine-tuning on the target task. The theory guides us to modify vanilla Transformer encoder architecture into Condenser. By actively conditioning on dense representation for MLM task, Condenser is readied for dense representation fine-tuning. Condenser demonstrates Optimization Ready is critical for more sample efficient and effective training of deep LM bi-encoder. For future research, our theory can provide guidelines on not only how to design more effective pre-training architecture but also how to further improve tasks like ICT to improve bi-encoder performance.

Our proposed architecture also has the potential for real-world application. For practitioners without massive compute resources, specialized pre-training architectures and tasks deviate the pre-trained model from BERT like general language understanding. When pre-training is not aligned well with the target task, they have to but may not not be able to pay the expensive pre-training cost. The fact that Condenser readies the model in the LM pre-training stage means most users can take pre-trained models directly into the fine-tuning pipeline and get instant benefits.

This paper studyies Optimization Readiness while assuming Knowledge Readiness can be established with large unsupervised corpus. Future works can explore setups where large unsupervised corpora are not naively attainable, to see how the two readiness issues interact.

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.

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.

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

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.

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. 2020a. 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.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, F. Petroni, V. Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

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.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *ArXiv*, abs/1910.05895.

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.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

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.

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.

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