Less is More: Pre-train a Strong Text Encoder for Dense Retrieval Using a Weak Decoder

Shuqi Lu[1*], **Di He**[2†], **Chenyan Xiong**[2†], **Guolin Ke**[2], **Waleed Malik**[2], **Zhicheng Dou**[1], **Paul Bennett**[2], **Tie-Yan Liu**[2], **Arnold Overwijk**[2]
[1]Renmin University of China [2]Microsoft
{lusq, dou}@ruc.edu.cn
{chenyan.xiong, dihe, guolin.ke, waleed.malik,
paul.n.bennett, tyliu, arnold.overwijk}@microsoft.com

Abstract

Dense retrieval requires high-quality text sequence embeddings to support effective search in the representation space. Autoencoder-based language models are appealing in dense retrieval as they train the encoder to output high-quality embedding that can reconstruct the input texts. However, in this paper, we provide theoretical analyses and show empirically that an autoencoder language model with a low reconstruction loss may not provide good sequence representations because the decoder may take shortcuts by exploiting language patterns. To address this, we propose a new self-learning method that pre-trains the autoencoder using a *weak* decoder, with restricted capacity and attention flexibility to push the encoder to provide better text representations. Our experiments on web search, news recommendation, and open domain question answering show that our pre-trained model significantly boosts the effectiveness and few-shot ability of dense retrieval models. Our code is available at https://github.com/microsoft/SEED-Encoder/.

1 Introduction

Recently, Dense Retrieval (DR) has progressed to more important roles in many language systems, for example, web search (Xiong et al., 2021), question answering (Karpukhin et al., 2020), and news recommendation (Wu et al., 2020b). In the first-stage retrieval of these scenarios, DR models generally employ a Siamese/Dual-Encoder architecture in practice. The encoder model first separately encodes the user side (query, browsing history, or question) and the corpus side (document or passages) as individual embeddings in a learned representation space (Lee et al., 2019), where retrieval with simple similarity metrics are conducted effectively (Johnson et al., 2017; Guo et al., 2020).

A popular choice of text encoders in DR is the Transformer network pre-trained by language modeling (e.g., BERT) (Reimers and Gurevych, 2019a). It is unexpected that, unlike in other language tasks where pre-trained models simply excel, directly fine-tuning BERT in DR often underperforms unsupervised sparse retrieval, e.g., BM25. Some complicated procedures are almost necessary to effectively fine-tune pre-trained Transformers in dense retrieval (Karpukhin et al., 2020; Luan et al., 2021; Xiong et al., 2021). One observation is that the pre-trained language models are not effective at encoding the semantics of the entire text sequence in one embedding, especially in dense retrieval where text sequences are mostly longer than 128 tokens (Luan et al., 2021).

In some other modalities, autoencoders have been widely used to obtain high-quality data representations (Vincent et al., 2010; Kingma and Welling, 2013). They pair a decoder on top of the encoder, trains the decoder to reconstruct the data solely from the encoder's encodings, thus enforce an information bottleneck on the data encodings for better representation quality. Recently, autoencoders have been brought in language pre-training. Li et al. (2020) stacks a GPT-2 decoder on top of the BERT encoder and trains the autoencoder via a conditional language modeling task. Their learned encoder, Optimus, provides better text encodings for GLUE and language generation tasks, but, as shown in our empirical study, does not provide better encodings for dense retrieval.

This phenomenon inspires us to investigate why the standard setup of autoencoders in language modeling falls short in dense retrieval. We first notice that in the auto-regressive decoder, the model takes not only the `CLS` encoding but also the previous tokens as input. Our mathematical analysis shows that the decoder can exploit natural language patterns using its access to previous tokens and bypass the dependency on the encoder, especially

[*]Work done while interning at Microsoft.
[†]Corresponding Authors.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2780–2791
November 7–11, 2021.

when the sequence is long and the decoder is strong, e.g., GPT-2. As a result, the autoencoder achieving a low reconstruction loss value does not necessarily provide better text sequence encodings.

Our analyses lead to a quite simple solution: we present a new autoencoder pre-training strategy, which pairs the BERT-style encoder with a weak decoder by restricting its parameter capacity and attention flexibility. This way, our SEED-Encoder, "Strong tExt Encoder by training with weak Decoder", creates an information bottleneck in the autoencoder and forces the encoder to provide better text representations. In our experiments on three real-world applications, we confirm that SEED-Encoder produces better pre-trained checkpoints that seed dense retrieval models with higher accuracy and better few-shot ability.

2 Related work

Pre-training Language Models. Masked Language Modeling (MLM) (Devlin et al., 2018) is one of the most effective ways to learn text representations. It first randomly masks some tokens in a sequence and then pre-trains a Transformer to recover them (Joshi et al., 2020; Liu et al., 2019; Clark et al., 2020). There are also attempts to design sequence-level tasks during pre-training. The next sequence prediction task proposed in Devlin et al. (2018) trains the model to predict whether two sequences are contiguous. Liu et al. (2019) showed this task is not effective and can be removed. In Sun et al. (2020), more sequence-level tasks are developed, such as predicting whether two segments are from the same document. Our learning framework architecture is close to Li et al. (2020), which trains an encoder and a decoder for both language understanding and generation. We will discuss its detail and show how it motivates our work.

Dense Retrieval with Text Encoders. Dense-Retrieval systems often use the Siamese/Dual Encoder architecture, where two sequences are encoded by the Transformer separately, and their similarity is calculated upon their sequence embeddings. Reimers and Gurevych (2019b) is among the first to study how to use BERT in a Siamese architecture and found that the `CLS` representation does not perform as well as expected. Recent research (Karpukhin et al., 2020; Xiong et al., 2021) demonstrated that applying pre-trained models in dense text retrieval is not as straightforward. Karpukhin et al. (2020) use BM25 to find negative samples to better fine-tune pre-trained models for dense retrieval. Xiong et al. (2021) performs global noise constructive estimation and finds global negatives using the DR model for the DR model.

3 Method

In this section, we first recap preliminaries in language pre-training and autoencoder. Then we discuss the drawbacks of using strong decoders in autoencoder and address them with SEED-Encoder.

3.1 Preliminary

In a standard setup of pre-training language models, e.g., BERT (Devlin et al., 2018), the neural network to be pre-trained is a multi-layer bidirectional Transformer encoder (Vaswani et al., 2017), which takes a sequence of tokens $x = (x_1, ..., x_n)$ from the vocabulary V, and produces their contextualized representations $\mathbf{h} = (\mathbf{h_1}, ..., \mathbf{h_n})$:

$$(\texttt{CLS}, x_1, ..., x_n) \xrightarrow{\text{Transformer}} (\mathbf{h_0}, \mathbf{h_1}, ..., \mathbf{h_n}),$$

where `CLS` is a special token added in the first position, its contextual representation $\mathbf{h_0}$ is often used as the representation of the sequence. The parameters of the Transformer θ_{enc} are typically pre-trained using Masked Language Modeling (MLM) (Devlin et al., 2018), which masks a fraction of the input sequence and trains the model to predict the original tokens. For ease of reference, we denoted the loss as $\mathcal{L}_{\text{MLM}}(x, \theta_{enc})$.

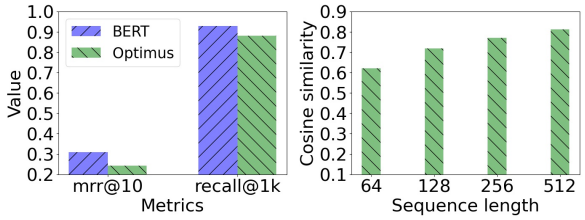
As there is no informative training target at the `CLS` position in token level pre-training tasks, it is not formally guaranteed that the contextual representation at `CLS` contains enough information for any sequence-level downstream tasks. Li et al. (2020) introduces the autoencoder setup in language model pre-training, which adds a reconstruction loss on top of the `CLS` token's $\mathbf{h_0}$:

$$x \xrightarrow{\theta_{enc}} \mathbf{h_0} \xrightarrow{\theta_{\mathbf{dec}}} \mathbf{x}. \tag{1}$$

where $\mathbf{h_0}$ is viewed as a latent variable. The decoder θ_{dec}, which is another deep Transformer model GPT-2, receives $\mathbf{h_0}$ and generates the original input autoregressively. The (variational) decoder loss is defined as (Li et al., 2020):

$$\mathcal{L}_{dec}(x, \theta_{dec}) = \\ -\sum_{t:1\sim n} \log P(x_t | x_{<t}, \mathbf{h_0}; \theta_{\mathbf{dec}}), \tag{2}$$

where $x_{<t}$ are all previous tokens before t.



(a) Ranking accuracy (b) Cosine similarity

Figure 1: Behaviors of Optimus on MS MARCO Passage Ranking Dev set: (a) its ranking accuracy in comparison with vanilla BERT; (b) its sequence representations' cosine similarity at variant lengths.

3.2 Effects of Using a Strong Decoder

One would expect the autoencoder to provide good representations if the decoder can well recover the input. However, we found that a typical model stacking a standard autoregressive decoder on a standard BERT-style encoder doesn't work well in dense retrieval tasks. For example, we fine-tune the pre-trained checkpoint of Optimus, which stacks GPT-2 on top of BERT on MS MARCO and compare it with BERT. We use Mean Reciprocal Rank(mrr) and recall as evaluation metrics. The detailed experimental setting can be found in Section 4.3, and the results are shown in Figure 1(a).

The performance of Optimus on dense retrieval tasks is worse than standard BERT, a sharp contrast with Optimus's effectiveness on other language tasks, e.g., in GLUE benchmarks. Note that one difference between data in GLUE and MS MARCO is the *sequence length*. In most GLUE tasks, the sequence length is short, e.g., average 14 tokens in SST-2, while the average passage length in MS MARCO is more than 450. Also, recent research shows that long sentences are hard to represent via single embedding vectors from pre-trained models (Luan et al., 2021).

To confirm this, We randomly select sequence pairs of different lengths and calculate the cosine similarity of their `CLS` embeddings provided by Optimus. The results are shown in Figure 1(b). The representations of long sequences (256 or 512 tokens) from Optimus are quite similar; the cosine similarities of random long sequence pairs are around 0.8. The model yields cluttered representations for long text sequences. When fine-tuned for dense retrieval in MS MARCO, it does not separate relevant documents for a query from those irrelevant ones. All of those representations might be similar to each other and require dedicated fine-tuning to realign their encodings.

3.3 Theoretical Analysis

Next, we mathematically show why the encoder may fail to learn good sequence representations using a strong decoder.

In Eqn. 2, at each time step t, the prediction of x_t not only depends on the `CLS` encoding $\mathbf{h_0}$ but also the previous tokens $x_{<t}$. Thus a lower reconstruction loss may not be contributed by more informative $\mathbf{h_0}$: for a large t in a long text sequence, the model may directly predict x_t from $x_{<t}$ if the decoder is strong. The quality of the representation at the `CLS` is not guaranteed as a low decoding loss may not reflect much about $\mathbf{h_0}$.

To further understand the requirements for informative sequence representations, we investigate the relationship between the reconstruction loss, $\mathbf{h_0}$, and the language sequence in their mathematical form. First, we decompose the expectation of the loss $\mathcal{L}_{dec}$ into two terms: a Kullback–Leibler divergence and a conditional-entropy term, according to the following fact in information theory:

Fact 1 *Given two distributions $P(Y,Z)$ and $Q(Y,Z)$ on random variables (Y, Z), we have*

$$\begin{aligned}&\mathrm{E}_{Y,Z\sim P}[-\log Q(Z|Y)]\\&=\mathrm{E}_{Y\sim P(Y)}[D_{KL}(P(Z|Y)||Q(Z|Y))]\\&\quad+H_P(Z|Y).\end{aligned}\tag{3}$$

We have X as a random variable defined in the sequence space $\mathcal{X}$, where each sequence x is sampled from data distribution P_D, $X_{<t}$ as the truncate of X at position t, and $P_{\theta_{dec}}$ as the sequence distribution generated by the decoder. For simplicity, we assume all the sequences are of length n. The expected reconstruction loss can be rewritten as

$$\mathrm{E}_D[\mathcal{L}_{dec}(X,\theta_{dec})] \tag{4}$$

$$=\mathrm{E}_D\left[\sum_{t:1\sim n}-\log P(X_t|X_{<t},\mathbf{h_0};\theta_{\mathbf{dec}})\right] \tag{5}$$

$$=\sum_{t:1\sim n}\mathrm{E}_D\Big[D_{\mathrm{KL}}\Big(P_D(X_t|X_{<t},\mathbf{h_0})|| \tag{6}$$

$$P_{\theta_{dec}}(X_t|X_{<t},\mathbf{h_0})\Big)\Big] \tag{7}$$

$$+H_D(X_t|X_{<t},\mathbf{h_0}). \tag{8}$$

The above equation shows that the loss consists of two terms, a K-L term $D_{\mathrm{KL}}(\cdot)$ (Eqn. 6 and Eqn. 7) describing the difference between two distributions, and a conditional-entropy term $H_D(\cdot)$ (Eqn. 8) reflecting the strength of language patterns. As we discuss next, both terms can achieve low values even with random $\mathbf{h_0}$.

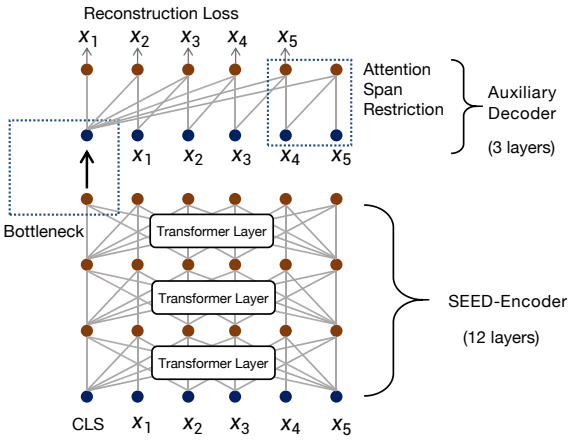


Figure 2: The structure of SEED-Encoder with an auxiliary decoder. The encoder and decoder are connected only via the [CLS] representation as the information bottleneck. The decoder capacity is restricted in both parameter size and attention span.

The first K-L term characterizes how $P_{\theta_{dec}}(X_t|X_{<t}, \mathbf{h_0})$, the decoder generated sequence distribution, aligns with the ground truth distribution $P_D(X_t|X_{<t}, \mathbf{h_0})$. Even with a meaningless θ_{enc}, if the decoder has sufficient capacity, e.g., a very deep Transformer, it can still approximate the ground truth distribution well and thereby reduce the K-L term. In theory, Transformers with arbitrary width and depth can approximate any sequence-level functions and may reach a low K-L loss using little information from $\mathbf{h_0}$ (Yun et al., 2019).

The second term $H_D(X_t|X_{<t}, \mathbf{h_0})$ characterizes the strength of language patterns: The stronger the correlation between X_t with $X_{<t}$, the lower the second term is. In natural language, the correlation becomes stronger with larger t as there is more information from the previous tokens. There is not a strong need for a good text encoder $\mathbf{h_0}$ because a strong decoder can capture the natural language patterns by itself.

3.4 SEED-Encoder

Our analysis shows that to obtain a stronger text encoder and a better $\mathbf{h_0}$, we can not make the decoder too strong: we need to constrain its capacity and also the available language context to reduce the correlation between X_t and $X_{<t}$, so that it has to rely on the information in the encoder `CLS` to reconstruct the text sequence.

In the rest of this section, We introduce SEED-Encoder which adopts these designs. The model structure is illustrated in Figure 2.

Making a language model weaker is easier than making it stronger. We simply modify Eqn. 2 to weaken the decoder:

- Using a shallower Transformer θ_{dec}^{weak} with fewer layers (e.g., three);
- Restricting its access to previous context, i.e., limit model attention to previous k tokens.

This leads to the following reconstruction loss:

$$\mathcal{L}_{dec}^{weak}(x, \theta_{dec}^{weak}) = - \sum_{t:1\sim n} \log P(x_t|x_{t-k:t-1}, \mathbf{h_0}; \theta_{\mathbf{dec}}^{\mathbf{weak}}), \quad (9)$$

where k is the window size of the restricted attention. Through these modifications, we enforce the information bottleneck between the encoder and the decoder, thereby forcing the decoder to rely on the `CLS` representation of the encoder, and pushing the encoder to learn a more informative representation.

Similar to Li et al. (2020), the pre-training of SEED-Encoder uses the combination of the encoder's standard MLM loss and the decoder's reconstruction loss:

$$\mathcal{L}(x, \theta_{enc}, \theta_{dec}^{weak}) = \mathcal{L}_{\text{MLM}}(x, \theta_{enc}) + \mathcal{L}_{dec}^{weak}(x, \theta_{dec}^{weak}). \quad (10)$$

The encoder and decoder are trained together. After pre-training, the decoder is discarded, and the encoder is used in downstream applications.

4 Experiments

In this section, we present various experimental analyses to evaluate the SEED-Encoder on dense retrieval tasks. More results on other language tasks are in Appendix A.2.

4.1 Pre-training Details

All our models are pre-trained *from scratch*, following the setup of BERT-base (Devlin et al., 2018): pre-training on English Wikipedia and BookCorpus (Zhu et al., 2015) (roughly 16GB texts) for 1M steps, with batch size 256, maximum sequence length 512, and 15% masks. We follow the pre-processing steps and use 32,768 sub-word tokens in Ke et al. (2020). We remove the next sentence prediction task following Liu et al. (2019).

	Rerank	Retrieval	
Model	**MRR@10**	**MRR@10**	**Recall@1k**
BM25 (Craswell et al., 2020)	-	0.240	0.814
Best DeepCT (Dai and Callan, 2019)	-	0.243	n.a.
Best TREC Trad IR (Craswell et al., 2020)	-	0.240	n.a.
DPR (RoBERTa) (Karpukhin et al., 2020)	-	0.311	0.952
With DPR (BM25 Neg)			
BERT (Devlin et al., 2018)	0.317	0.310	0.929
Optimus (Li et al., 2020)	0.300	0.244	0.880
ELECTRA (Clark et al., 2020)	0.300	0.258	0.854
ERNIE2.0 (Sun et al., 2020)	0.324	0.321	0.942
RoBERTa (Liu et al., 2019)	-	0.299	0.928
BERT (Ours)	0.326	0.320	0.933
SEED-Encoder	**0.329**†	**0.329**†	**0.953**†
With ANCE (FirstP)			
RoBERTa (Liu et al., 2019)	-	0.330	0.959
BERT (Ours)	0.327	0.332	0.952
SEED-Encoder	**0.334**†	**0.339**†	**0.961**†

Table 1: First stage retrieval results on MS MARCO Passage ranking Dev set. Rerank MRR is for reference only. Statistically significant improvements over BERT (Ours) are marked by †.

We use Adam (Kingma and Ba, 2014) as the optimizer, and set its hyperparameter ϵ to 1e-6 and $(\beta 1, \beta 2)$ to (0.9, 0.999). The peak learning rate is set to 1e-4 with a $10k$-step warm-up stage. After the warm-up stage, the learning rate decays linearly to zero. We set the dropout probability to 0.1, gradient clip norm to 1.0, and weight decay to 0.01. All codes are implemented based on *fairseq* (Ott et al., 2019) in *PyTorch* (Paszke et al., 2017). All models are run on 8 NVIDIA Tesla V100 GPUs with mixed-precision (Micikevicius et al., 2017).

Our encoder architecture is the same with BERT-base: 12 Transformer layers, eight attention heads, and 768 hidden dimensions (110M parameters). We use a three-layer Transformer as the decoder, restrict its attention to the previous two tokens (attention span $k = 2$), and keep all else the same with the encoder. The decoder is only used in pre-training and is dropped during fine-tuning. There is no additional cost in fine-tuning nor inference.

4.2 Fine-tuning Siamese/Dual-Encoders

Fine-tuning SEED-Encoder in the Siamese architecture on the dense retrieval tasks is the same as other pre-trained models. Here we show how fine-tuning in a typical sentence pair matching task with binary labels can be done with Triplet loss.

$$\mathcal{L} = \sum_{x^q, x^{d+}, x^{d-}} \text{relu}(1 - (s(x^q, x^{d+}) - s(x^q, x^{d-}))).$$

The training data include triples of query x^q and its positive/negative labeled sequence (x^{d+}, x^{d-}). The scoring of the sequence pairs $s(x^q, x^d)$ is done by simple similarity functions, such as cosine and dot product, on their `CLS` encodings. More advanced fine-tuning strategies (Karpukhin et al., 2020; Xiong et al., 2021) can also be used as SEED-Encoder is an alternative for other pre-trained encoders.

4.3 Experiments on Web Search

Our first application, web search (Lee et al., 2019),.uses the MS MARCO (Bajaj et al., 2016) dataset, the largest public search benchmark to date. It includes two tasks, passage ranking and document ranking. We focus on the first stage retrieval step, which is to find relevant passages/documents from the entire corpus. We also show the results in the reranking setting where all models rerank a pre-given set of candidate documents (Top 100 from BM25) for reference. More details of MARCO are in Appendix A.1.

Our pre-trained encoders are fine-tuned with ANCE negative sampling strategy (Xiong et al., 2021). In document retrieval, we use ANCE (FirstP) which uses the first 512 tokens of the long document and cut-off the rest. We also evaluate with another negative sampling strategy, BM25 Neg, which uses top 100 BM25 retrieved results as negatives samples and performs similar to DPR (Karpukhin et al., 2020) on MARCO.

Baselines: The main baseline is our run of BERT-base (Devlin et al., 2018; Liu et al., 2019), which we pre-trained and fine-tuned in the exact setting with SEED-Encoder. We use the permutation test and $p < 0.05$ as the statistical significance test between SEED-Encoder and BERT (Ours). Besides BERT, we evaluate two other pre-trained language models in the same setting: ELECTRA (Clark et al., 2020) and ERNIE2.0 (Sun et al., 2020). ELECTRA is one of the most effective pre-trained encoders on the GLUE benchmark (Clark et al., 2019). ERNIE2.0 uses various token-level tasks and sentence-level tasks, including an IR Relevance Task. We use the MARCO passage benchmark to showcase the performance of these two pre-trained models.

In addition, we also list the task-specific first stage retrieval baselines that were published recently or submitted to the leaderboard, although they barely outperform our vanilla BERT baseline. For passage ranking, the classic sparse retrieval baselines include the standard BM25, Best TREC Sparse Retrieval with tuned query expansion, and Best DeepCT, all from TREC DL 2019 official evaluation (Craswell et al., 2020). These three approaches represent the standard sparse retrieval,

Model	Dev		Eval
	Rerank	Retrieval	Retrieval
BM25 (Craswell et al., 2020)	-	0.318	0.284
DE-hybrid (Luan et al., 2021)	-	-	0.287
BM25 + doc2query-T5 expansion	-	0.327	0.291
ME-hybrid (Luan et al., 2021)	-	-	0.310
Enriched Traditional IR Baseline	-	0.355	0.312
ANCE MaxP (RoBERTa) (Xiong et al., 2021)	-	0.384	0.342
With DPR (BM25 Neg)			
BERT (Ours)	0.338	0.308	-
SEED-Encoder	**0.344**†	**0.323**†	-
With ANCE (FirstP)			
RoBERTa (Liu et al., 2019)	-	0.373	-
BERT (Ours)	0.368	0.382	-
SEED-Encoder	**0.377**†	**0.394**†	**0.362**

Table 2: MRR@100 on MARCO Documents from first-stage retrieval methods. Rerank results are for reference only. Statistically significant improvements over BERT (Ours) are marked by †.

best classical sparse retrieval, and the latest method of using BERT to improve sparse retrieval.

For document ranking, BM25 (Craswell et al., 2020) and the enriched traditional IR baseline are standard sparse retrieval baselines. The enriched traditional IR baseline uses pre-defined IR features, including BM25, to rank the documents. BM25 + doc2query-T5 expansion uses Doc2query model(Nogueira et al., 2019), expanding the documents with predicted queries that are related to or representative of the documents' content. The queries are predicted by a sequence-to-sequence model taking the document terms as input. Both DE-hybrid and ME-hybrid (Luan et al., 2021) use dense features from BERT and hand-craft sparse features. DE-hybrid takes the *CLS* representations of document and query as the dense feature and calculates the dot product similarity. This similarity score is further combined with sparse retrieval scores as the final score for ranking. Different from DE-hybrid, ME-hybrid uses max-pooling over multiple contextual embeddings as dense features.

Results: The results of SEED-Encoder and baselines in MARCO Passage retrieval and Doc retrieval are listed in Table 1 and Table 2. SEED-Encoder outperforms all existing baselines on all benchmarks. By simply switching its fine-tuning starting checkpoint from BERT to SEED-Encoder—without changing any architectures nor fine-tuning strategies—the accuracy is significantly improved on these two large-scale benchmarks.

In comparison, on MARCO Passage retrieval, switching from BERT to ELECTRA or ERNIE2.0 does not improve the retrieval accuracy. Pre-training models optimized for other scenarios are not necessarily better for dense retrieval.

On MARCO document retrieval, ANCE (FirstP) only uses one vector per document from its first passage, while ANCE (MaxP) uses four vectors per document from its first four passages, which often cover the full document body. Yet with SEED-Encoder as the starting point, ANCE (FirstP) outperforms the recent state-of-the-art ANCE (MaxP) with RoBERTa by relatively 6% in the hidden Eval, while using fewer embeddings per document. Reducing embeddings required per document is important in real search systems where the corpus size is beyond billions (Xiong et al., 2021).

Model	AUC	MRR	NDCG@5	NDCG@10
Transformer (Vaswani et al., 2017)	0.6776	0.3305	0.3594	0.4163
Transformer-XL (Dai et al., 2019)	0.6792	0.3315	0.3604	0.4170
TENER (Yan et al., 2019)	0.6770	0.3301	0.3589	0.4158
DA-Transformer (Wu et al., 2020a)	0.6832	0.3336	0.3634	0.4207
With DPR (MIND Neg)				
BERT (ours)	0.7015	0.346	0.3844	0.4479
SEED-Encoder	**0.7059**†	**0.3506**†	**0.3908**†	**0.4526**†

Table 3: Results on MIND news recommendation benchmark. All methods are evaluated in the reranking setting with pre-given news candidates in MIND, to follow their official setting. Baseline scores are obtained from Wu et al. (2020a). Statistically significant improvements over BERT (Ours) are marked by †.

4.4 Experiments on News Recommendation

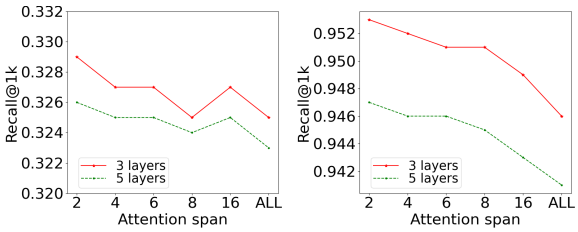
Our second application is news article recommendation, another important real-world task that connects users with information. We use the recently released MIcrosoft News Dataset (MIND) benchmark (Wu et al., 2020b). The task is to rank a given set of candidate news articles based on the user's previous click history on MSN news articles. The evaluation uses the user's click as the positive label. We use the public MIND Dev and its official metrics: AUC, MRR, NDCG@5, and NDCG@10. More details of MIND are in Appendix A.1.

We follow MIND's official setting and use a standard dense retrieval model to rerank the pre-given candidate news articles. Our DR model represents each user's history by concatenating all the titles they clicked on the MSN site, with [SEP] tokens in between, and using as many recent titles as possible within the 512 length limit. The candidate articles are represented by the concatenation of their titles and snippets. Then it encodes the user history and candidate articles with SEED-Encoder, and matches them with dot-products.

Baselines: MIND is a relatively new benchmark. The most recent baselines are those in Wu et al. (2020a). Based on Transformer (Vaswani et al., 2017), Transformer-XL (Dai et al., 2019) uses relative positional encodings that integrate content-dependent positional scores and a global positional

Model	Top-20	Top-100
BM25 (Craswell et al., 2020)	59.1	73.7
With DPR		
BERT (Karpukhin et al., 2020)	78.4	85.4
BERT (BM25 +DPR) (Karpukhin et al., 2020)	76.6	83.8
BERT (Ours)	77.8	85.1
SEED-Encoder	**80.4**†	**87.1**†
With ANCE		
BERT (Xiong et al., 2021)	81.9	87.5
SEED-Encoder	**83.1**†	**88.7**†

Table 4: Retrieval results (Answer Coverage at Top-20/100) on Natural Questions in the setting from (Karpukhin et al., 2020). Statistically significant improvements over BERT are marked by †.



(a) MRR@10 (b) Recall@1k

Figure 3: MS MARCO passage Dev accuracy of Siamese (BM25 Neg) when fine-tuned from SEED-Encoder variations.

score in the self-attention layer. TENER (Yan et al., 2019) uses direction-aware sinusoidal relative position embeddings in a similar way as in Transformer-XL. Different from Transformer-XL and TENER, DA-Transformer (Wu et al., 2020a) directly rescales the attention weights based on the mapped relative distances instead of using sinusoidal position embeddings. Similar to the web search experiments, we also compare SEED-Encoder with BERT (Ours).

Results: The results of SEED-Encoder and baselines in MIND are listed in Table 3. SEED-Encoder outperforms the recent state-of-the-art DA-Transformer, which employs various architecture improvements specifically designed for recommendation (Wu et al., 2020a). A better self-learning strategy to leverage unsupervised data can be as effective as, if not better than, task-specific architecture changes while avoiding all the engineering hustles.

4.5 Experiments on Open QA

Our third application is dense retrieval in open-domain question answering. This task often leverages a two-stage framework: first uses a context retriever to select a small set of passages that may contain the answer to the question; and then uses a machine reader which thoroughly examines the retrieved passages and identifies the correct answer (Karpukhin et al., 2020). We focus on the first stage, i.e., dense retrieval to select relevant passages. We use Natural Question query set (Kwiatkowski et al., 2019) and the Wikipedia passages prepared and shared in DPR (Karpukhin et al., 2020). More details of the NQ dataset are in Appendix A.1. We follow the evaluation metrics used in DPR, hit accuracy of Top-20 and Top-100.

Models are fine-tuned using DPR fine-tuning strategy as in Karpukhin et al. (2020), which uses a dual-encoder architecture and samples negatives in the mini-batch. We also experiment with the ANCE fine-tuning strategy as in Xiong et al. (2021) which dynamically samples hard negatives.

Baselines: We take BM25, BERT as baselines as in Karpukhin et al. (2020). Consistent with the web search tasks and news recommendation tasks, we also compare SEED-Encoder with BERT (ours).

Results: The results of SEED-Encoder and the baselines on NQ benchmark are in Table 4. Again, SEED-Encoder outperforms all other baselines with DPR negative sampling or ANCE negative sampling. We do not change any architectures nor fine-tune strategies and only simply switch the BERT checkpoint to SEED-Encoder, but bring significant improvements on the large-scale benchmark.

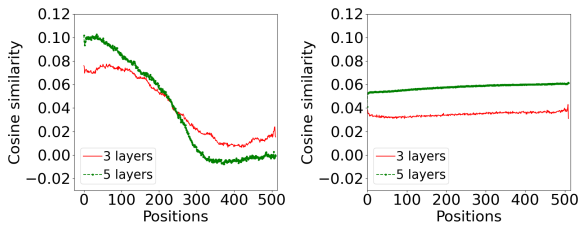
4.6 Discussion and Analysis

In this section, we conduct more analysis to understand the advantages of the SEED-Encoder. For simplicity, all experiments are run on the MS MARCO document retrieval tasks.

4.6.1 Ablation study

In the experiments above, we use a three-layer Transformer decoder and restrict the attention span to be two. One may wonder whether such constraints are essential for learning good sentence representations. In this section, we try various decoder configurations with different numbers of layers and attention window sizes.

From the results in Figure 3, we can see that the SEED-Encoder with the stronger decoder, 5-layer Transformer with full attention (All), performs worse than those with weaker decoders in dense retrieval. The retrieval accuracy correlated well with the decoder capacity of the corresponding SEED-Encoder. So unlike typical multi-task settings where tasks share lower-level representa-



(a) Full Attention. (b) Restricted Attention.

Figure 4: The cosine similarity between encoder CLS and the token representations from the decoder at different positions: 0 is the beginning of the sequence and the closest to CLS. The restricted attention sets attention span to two.

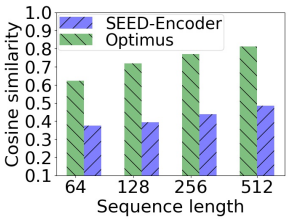


Figure 5: Cosine similarity of sequences with different lengths using Optimus and SEED-Encoder.

tions and correlate in accuracy, in SEED-Encoder, the decoder is to force the encoder to capture more information in its sequence embeddings: A weak decoder leads to a stronger encoder.

To further understand the relationship of encoder's CLS embedding and the decoder, in Figure 4 we plot the cosine similarity between the decoder's token representations in its last layer and the encoder's CLS. The impact of restricting attention is significant: with full attention (Figure 4(a)), the decoder may depend heavily on the encoder's CLS in the beginning but quickly drops the dependency when sufficient context information is available; with restricted access to context, the decoder is forced to attend more on the encoder's CLS representation in all token positions, as shown in the consistent cosine similarity in different positions in Figure 4(b). This confirms that when the decoder is weak (restricted attention), it depends more on the encoder's CLS, thus pushes the encoder to learn more informative representations. Also, the results in Figure 4(a) suggest that when using a powerful encoder, the CLS embedding will encode the first several words in the sentence but might ignore others. This can be one of the reasons that Optimus performs worse than BERT in dense retrieval in Figure 1(a).

4.6.2 Document Representation Quality

In Section 3.2, we empirically show that using a standard autoencoder learning framework, the similarity between sequence representations grows to be large for long sequences. In this section, we first study whether SEED-Encoder improves the representation diversity. Similar to Figure 1(b), we collect randomly sampled sentence pairs and calculate the cosine similarity of their CLS encodings generated by SEED-Encoder.

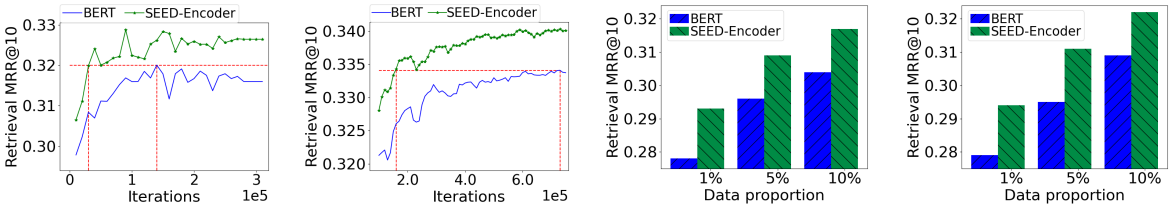
Results in Figure 5 shows that, the CLS embedding generated by SEED-Encoder is more diverse. The average CLS cosine similarity is only 0.48 even when the sentence length is 512. This result shows that SEED-Encoder can well differentiate sentences during pre-training.

Few-shot effectiveness Note that diverse representations don't necessarily mean high-quality. To figure out the effectiveness of the representation, we conduct few-shot learning experiments for SEED-Encoder. In particular, we record the dev performance during the fine-tuning stage and check how many training iterations and how many samples are required for the model to achieve a reasonably good performance.

In Figure 6(a) and 6(b), we plot the retrieval accuracy at different fine-tuning steps. Starting from SEED-Encoder instead of BERT, both the vanilla Siamese and ANCE achieve higher retrieval accuracy in the very beginning and maintain their advantages throughout the fine-tuning process. For example, Siamese (BM25 Neg) only requires 30k fine-tuning iterations with SEED-Encoder to reach BERT's best performance at 140k iterations. With ANCE (First P), it takes 150K iterations with SEED-Encoder versus 750K with BERT.

In Figure 6(c) and 6(d), we plot the retrieval accuracy with different fractions of training data. Compared with BERT, with fewer training labels, SEED-Encoder always reaches better accuracy. When only using 10% training labels, SEED-Encoder (MRR 0.318 in Figure 6(c)) is still competitive with BERT using all training labels (MRR 0.32).

These results indicate that the representation learned by SEED-Encoder is better than that learned by BERT. The reduction in fine-tuning cost helps democratize the benefits of pre-training models, especially in applications where computing resources or task-specific supervision is restricted.



(a) Siamese (BM25 Neg). (b) ANCE (FirstP). (c) Siamese (BM25 Neg). (d) ANCE (FirstP).

Figure 6: MS MARCO passage retrieval accuracy of Siamese (BM25 Neg) and ANCE (FirstP) when fine-tuned from BERT (Ours) and SEED-Encoder. (a) and (b) are their accuracy at different fine-tuning steps (x-axes, in 100K). (c) and (d) are their accuracy with a fraction (x-axes) of training labels in the few-shot setting.

	Case 1	**Case 2**
Query	hiking on mount rainier in the winter	what kind of party is the cooperative party
SEED-Encoder	MRR@100 1.0	MRR@100 1.0
Url	https://www.nps.gov/mora/planyourvisit/winter-recreation.htm	https://simple.wikipedia.org/wiki/Co-operative_Party
Title	Winter Recreation	Cooperative Party
Snippet	Winter Recreation Winter Camping Food Storage Snowplay... **A Winter visit to Mount Rainier** can include ranger-guided snowshoe walks, skiing...Learn about **winter hiking** opportunities at Longmire in...	**Co-operative Party** From Wikipedia, the free encyclopedia navigation search. The **Co-operative Party** is a small socialist political party, in the United Kingdom. Its candidates must be members of the Labour Party as well...
RoBERTa	MRR@100 0.043	MRR@100 0.067
Url	http://www.seattletimes.com/life/travel/5-great-day-hikes-around-mount-rainier/	http://socialeconomyaz.org/whats-a-cooperative/
Title	5 great day-hikes around Mount Rainier	What is a Cooperative?
Snippet	Life Outdoors Travel5 great day-hikes around Mount Rainier Originally published June 24, 2015 at 4:59...(Picasa)E-book authors name their favorite day-hikes in Mount Rainier National Park...	What is a Cooperative? According to the International Cooperative Alliance (ICA), a cooperative is "an autonomous association of persons united voluntarily to meet their common economic, social, and cultural needs...

Table 5: Two examples of SEED-Encoder's winning case over RoBERTa (Ours) when fine-tuning with ANCE FirstP in MARCO Document. Their first ranked documents are listed.

Case Study We further showcase some winning examples of SEED-Encoder in Table 5. The error made by BERT correlated with our observation in Figure 4(a), where the encoder's representation is more related to those tokens at the beginning of the text sequences, which is quite related to the query. Only when the model captures the information of the entire text can it find the correct documents. For example, in the first case, SEED-Encoder captures "winter hiking" at the back of the document while BERT only pays attention to some of the keywords at the beginning of the document even if the overall semantics does not match, and in the second case, BERT missed the "party" part in the query.

5 Conclusion

In this paper we present SEED-Encoder, a self-training framework dedicated to pre-training language models for dense text retrieval. We pre-train an auto-encoder that employs a weak decoder with restricted capacity and attention span following our mathematical derivation. The weak decoder helps SEED-Encoder capture more context information and generate better text representation. In our experiments on web search, news recommendation, and question answering, SEED-Encoder initialized dense retrieval models achieve state-of-the-art accuracy compared to several strong baselines. Future work along this direction includes exploring more self-learning tasks and network architectures for sequence matching in dense retrieval scenarios.

Acknowledgements

We would like to thank anonymous reviewers for their valuable comments. This work is partially supported by National Natural Science Foundation of China NO. 61872370, and Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder,

Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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*, pages 3887–3896. PMLR.

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

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Guolin Ke, Di He, and Tie-Yan Liu. 2020. Rethinking the positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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*, page 6086–6096.

Chunyuan Li, Xiang Gao, Yuan Li, Xiujun Li, Baolin Peng, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. *arXiv preprint arXiv:2004.04092*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329––345.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12).

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2020a. Da-transformer: Distance-aware transformer. *arXiv preprint arXiv:2010.06925*.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020b. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606.

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

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. Tener: Adapting transformer encoder for named entity recognition. *arXiv preprint arXiv:1911.04474*.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. 2019. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

	Document			Passage		
Model	**Train**	**Dev**	**Eval**	**Train**	**Dev**	**Eval**
Query	367,013	5,193	5,793	808,731	101,093	101,092
relevant label	384,597	5,478	-	532,761	59,273	-
Doc set	3,213,835			8,841,823		

Table 6: Statistics of the MSMARCO dataset

	Train	**Dev**
Users	711,222	255,990
News	101,527	72,023
Impression	2,232,748	376,471
Avg. title len	14.41	14.47
Avg. click num	1.52	1.53
Avg. candidate num	37.40	37.41
Avg. historical news click num	32.98	32.62

Table 7: Statistics of the MIND dataset

A Appendix

A.1 More Details of MS MARCO, MIND and OpenQA dataset

More Details of MARCO Dataset Microsoft MARCO (Bajaj et al., 2016) is the largest available search benchmark to date. It includes two tasks, document ranking and passage ranking. Both are to find and rank relevant documents/passages from a web corpus for a web query from Bing. The dataset statistics are summarized in Table 6.

More Details of MIND Dataset MIcrosoft News Dataset (MIND) (Wu et al., 2020b) is a large-scale recommendation dataset that collects about 160k English news articles and more than 15 million user impression logs from MSN news. Each news article contains the title, abstract, body, and category. Each impression log includes the user's click behavior on the page and her historical news click behaviors. The task is to rank a given set of candidate news articles, e.g., those from an early stage of their recommendation pipeline, based on the user's previous click history. The dataset statistics are summarized in Table 7.

More Details of NQ Dataset For OpenQA experiments we use the Natural Question query set (Kwiatkowski et al., 2019), in which the queries are mined from real Google search queries and the corresponding answers are spans in Wikipedia articles identified by annotators. We use the Wikipedia passages preprocessed and shared in DPR (Karpukhin et al., 2020), which includes $21,015,324$ passages. More detailed data such as the number of queries can be found in Karpukhin et al. (2020)

model	MNLI	QQP	SST-2	QNLI
BERT (Ours)	0.849	0.910	0.929	0.913
Optimus	0.834	0.909	0.923	0.912
SEED-Encoder	0.843	0.911	0.927	0.914

Table 8: Results on some GLUE tasks.

A.2 GLUE

We also consider the GLUE benchmark (Wang et al., 2018) which contains nine datasets for general language understanding. Here we select MNLI, QQP, QNLI and SST-2 from the GLUE benchmark, and compare the performance of SEED-Encoder with BERT (Ours) and Optimus on these tasks. We follow the fine-tuning schedule in Devlin et al. (2018), and the results are shown in Table 8. We can see that on these GLUE tasks, SEED-Encoder is not worse than BERT and Optimus. This shows that while SEED-Encoder can generate higher-quality representations that well fit the Siamese network, the performance on GLUE will not become worse.