You Only Need One Model for Open-domain Question Answering

Haejun Lee[♣] **Akhil Kedia**[♣] **Jongwon Lee**[♢*] **Ashwin Paranjape**[♠]
Christopher D. Manning[♠] **Kyoung-Gu Woo**[♡*]
♣ Samsung Research ♢ Samsung Electronics
♠ Stanford University ♡ Growdle Corporation
{haejun82.lee, akhil.kedia, jay722.lee}@samsung.com
{ashwinp, manning}@cs.stanford.edu, epigramwoo@growdle.com

Abstract

Recent approaches to Open-domain Question Answering refer to an external knowledge base using a retriever model, optionally rerank passages with a separate reranker model and generate an answer using another reader model. Despite performing related tasks, the models have separate parameters and are weakly-coupled during training. We propose casting the retriever and the reranker as internal passage-wise attention mechanisms applied sequentially within the transformer architecture and feeding computed representations to the reader, with the hidden representations progressively refined at each stage. This allows us to use a single question answering model trained end-to-end, which is a more efficient use of model capacity and also leads to better gradient flow. We present a pre-training method to effectively train this architecture and evaluate our model on the Natural Questions and TriviaQA open datasets. For a fixed parameter budget, our model outperforms the previous state-of-the-art model by 1.0 and 0.7 exact match scores.

1 Introduction

Open-domain Question Answering (Open QA) is a knowledge-intensive task that finds the answer for the given question from a large-scale knowledge corpus that can easily surpass millions of documents. Thus, how to store and refer to the knowledge at such scales is important in terms of both performance and scalability for Open QA systems. Traditional systems rely on information retrieval engines such as Lucene. These score the relevance of knowledge to a given query by lexical overlaps between them in a sparse representation space based on TF-IDF or BM25 (Chen et al., 2017; Wang et al., 2018; Yang et al., 2019). However, recent advances in neural language modeling have enabled two new lines of approach; 1) referring to internal knowledge parameterized in the model (Brown et al., 2020; Petroni et al., 2019; Roberts et al., 2020), and 2) referring to external knowledge retrieved by matching query and knowledge in dense representation spaces (Karpukhin et al., 2020; Lee et al., 2019; Guu et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021b).

Despite the simplicity of the approach, parametric models have limitations such as a large number of model parameters that require large compute for both training and inference and non-expandable knowledge without re-training. Their implicit knowledge reference also makes it hard to find supporting knowledge and often results in hallucinations (Shuster et al., 2021). The current dense retrieval models have advantages over parametric models on these issues (Karpukhin et al., 2020; Guu et al., 2020; Lewis et al., 2020). But most retrieval models only have a weak coupling between and separate parameters for the reader, reranker (if any), and retriever that limits these models from achieving optimal end-to-end training and efficient use of the total model capacity.

In this paper, we propose a single language model YONO (You Only Need One model) that can refer to external knowledge via its internal attention functions, which are trainable in a fully end-to-end manner. We achieve this by generalizing the retrieval and reranking as internal passage-wise attentions. At the lower retrieval layers, the query and passages are separately encoded allowing pre-computation of all the passage representations. Then passage-wise hard attention is applied to retrieve initial relevant passages from the entire knowledge base. While it would be optimal to retrieve passages based on cross-attention between the query and all the passages (Khattab and Zaharia, 2020), it is computationally intractable. Hence, we approximate this attention by a passage-wise hard-attention layer using decoupled query

[*]Work done while at Samsung Research

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3047 - 3060
December 7-11, 2022

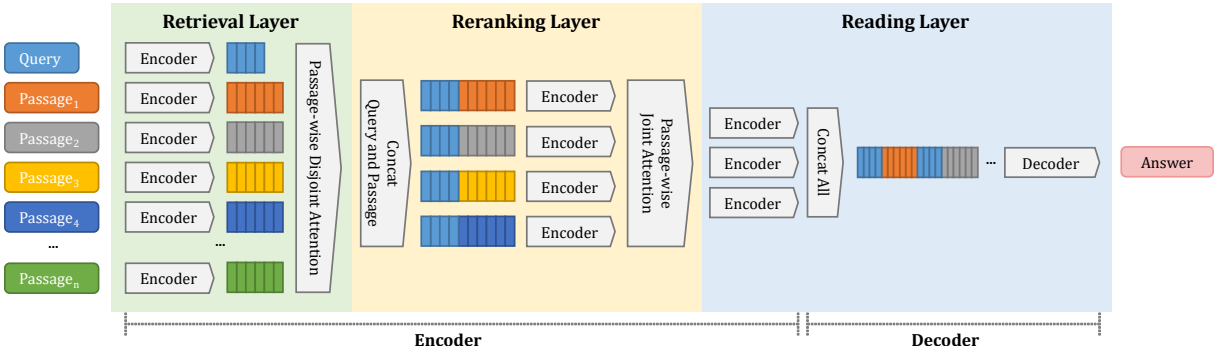

Figure 1: The overall architecture of our proposed model YONO.

and passage representations. The representations of the initial relevant passages are further encoded jointly with the query representation to compute more expressive coupled representations. These are used to select only the more relevant passages using another passage-wise hard-attention in the reranking layer. The representations of the final set of passages are then encoded by transformer encoders for deeper representations that are fused in the decoder to generate the answers.

We train this architecture fully end-to-end by self-supervised pre-training and weakly supervised fine-tuning without passage labels.

Our contributions are twofold;

- A single model that generalizes retrieval, reranking, and reading as internal attention functions. We show that this model trained end-to-end significantly improves the retrieval performance by leveraging a training signal from the answer generation decoder to allow better gradient flow across the whole model in Section 6.1. It also achieves better utilization of the model parameters, outperforming a stand-alone reader with the same number of parameters by 7.1% and 3.2% on NQ and TQA respectively as shown in Section 6.2.

- A method to train this architecture in a fully end-to-end manner. We show our pre-training method requires 51.5% fewer pre-training tokens compare to the previous the state-of-the-art approach in Section 7.2.

2 YONO Architecture

As depicted in Figure 1, we propose a single encoder-decoder language model architecture consisting of 3 components: Retrieval Layer, Reranking Layer, and Reading Layer.

2.1 Retrieval Layer

This first layer retrieves the top-N relevant passages for a given query from the knowledge corpus using query and passage representations independently encoded with the first K transformer encoder layers. The query is encoded with '*query*:' prefix while passages are encoded with '*title*:' and '*context*:' prefixes following previous approaches (Izacard and Grave, 2021b; Singh et al., 2021).

Passage-wise Disjoint Attention: Let q_0 and P_0 be the first tokens' representations of the query and all the passages respectively encoded independently. The disjoint attention scores are calculated by the scaled dot-product attention scores (Vaswani et al., 2017) between q_0 and P_0 as:

$$Q = LayerNorm(q_0 W_q)$$
$$K = LayerNorm(P_0 W_p)$$
$$score_{disjoint}(q, P) = \sigma(QK^T/\sqrt{d_k}) \quad (1)$$

where $W_q, W_p \in \mathrm{R}^{\mathrm{d}\times\mathrm{d}}$ are learned linear projections and $1/\sqrt{d_k}$ is the scaling factor following Vaswani et al. (2017).

The top-N relevant passages P^R with the highest $score_{disjoint}$ for a given query are selected and passed to the next layer. In practice, we retrieve top-N passages by indexing the pre-computed passage representations P_0 using Maximum Inner Product Search tools (MIPS) such as FAISS (Johnson et al., 2021). During training, the index is iteratively refreshed by the most recent model's representation following other neural retrieval approaches (Karpukhin et al., 2020; Lee et al., 2019; Guu et al., 2020; Izacard and Grave, 2021b).

2.2 Reranking Layer

We further narrow down the retrieved passages by applying an additional passage-wise attention

based on more expressive representations from the joint encoding of query and passages.

Passage-wise Joint Attention: We concatenate query and retrieved passage representations and encode them with cross-attention for more expressive representations using the next L transformer encoder layers. Let h^n be the encoded representations of query and the n^{th} passage and H_0 be the first token's representations of all encoded representations. We apply the second passage-wise attention based on $score_{joint}(q, P^R)$, obtained from H_0:

$$h^n = Transformer(q \oplus p^n) \quad (2)$$
$$H^0 = [h_0^0, h_0^1, h_0^2, ..., h_0^{N-1}]$$
$$score_{joint}(q, P^R) = \sigma(LayerNorm(H_0)W_{qp})$$

where $\oplus$ is the concatenation operation and $W_{qp} \in \mathrm{R}^{\mathrm{d}\times 1}$ is a learnt vector.

2.3 Reading Layer

After the retrieval and reranking layers, the final representations are fed to the reading layer. These are further encoded using the remaining transformer encoder layers and fused in the decoder for multi-passage reading, following the approach of FiD (Izacard and Grave, 2021b).

3 Training YONO

3.1 Training Objective

The whole model is always trained end-to-end by leveraging a training signal from the final answer generation. Due to non-differentiability of the passage-wise attentions, we combine additional losses $\mathcal{L}_{retrieval}$ and $\mathcal{L}_{reranking}$ to the answer generation $\mathcal{L}_{reading}$ as below:

$$\mathcal{L} = \mathcal{L}_{retrieval} + \mathcal{L}_{reranking} + \mathcal{L}_{reading} \quad (3)$$

Retrieval and Reranking Loss: The passage-wise attention scores $S_{retrieval}$ and $S_{reranking}$ for retrieval and reranking layers are calculated by Equation (1) and (2) using retrieved passages P^R and in-batch negative passages P^N as:

$$S_{retrieval} = score_{disjoint}(q, P^R \cup P^N)$$
$$S_{reranking} = score_{joint}(q, P^R) \quad (4)$$

In-batch negative passages P^N are used to expand $S_{retrieval}$ for more contrastive training signals. In-batch negatives not used for $S_{reranking}$ because the joint representation is only calculated for the retrieved passages, not the in-batch negatives.

These attention scores are not differentiable by the reader's generation loss because they are only used to select top-N passages at retrieval and reranking layers but not directly used in the answer generation. Instead, we train $score_{retrieval}$ and $score_{reranking}$ to approximate the target scores, which following previous work (Izacard and Grave, 2021a) are derived by accumulating the decoder's attention scores across decoder layers and attention heads over all encoded passage tokens as:

$$score_{decoder}(P) = \quad (5)$$
$$\sum_{l=0}^{N_l} \sum_{h=0}^{N_h} \sum_{t_p=0}^{N_t^p} \frac{SG(att_{dec}(0, l, h, t_p))}{N_l N_h N_t^p} \mid p \in P$$

where att_{dec} is decoder attention matrices toward encoded outputs, 0 is an output token index, N_l is the number of layers, N_h is the number of attention heads, N_t^p is the number of tokens in a given passage, and SG is a stop gradient function. The gradient flow back to the decoder's attention scores is blocked by SG to train the decoder by $\mathcal{L}_{reading}$ only.

Using this scoring function, we get target scores $T_{retrieval}$ and $T_{reranking}$. The scores of in-batch negative passages P^N are set to 0.

$$T_{retrieval} = score_{decoder}(P^R) \oplus (0 \mid p \in P^N)$$
$$T_{reranking} = score_{decoder}(P^R) \quad (6)$$

Finally, the losses for training passage-wise attention scores are obtained by KL-Divergence between target scores T and attention scores S as:

$$\mathcal{L}_{retrieval} = D_{KL}(T_{retrieval} \| S_{retrieval})$$
$$\mathcal{L}_{reranking} = D_{KL}(T_{reranking} \| S_{reranking}) \quad (7)$$

In addition, we add a constant penalty γ to $score_{retrieval}$ of in-batch negative passages P^N before applying a softmax of Equation (1). This inductive bias enforces the model to further decrease scores of the random negative passages lower than the lowest score of the retrieved passages.

Reading Loss: We use a conventional auto-regressive language modeling loss for generating an answer a given a query q and retrieved passages P^R:

$$\mathcal{L}_{reading} = -\log \prod_{t=1}^{T_A} p(a_t \mid a_{<t}, q, P^R) \quad (8)$$

3.2 Pre-training Corpus

We first pre-train our model to adapt the pre-trained encoder-decoder architecture to the YONO architecture and provide initial retrieval performance for fine-tuning without passage labels. Inverse Cloze Task (ICT) (Lee et al., 2019) and Masked Salient Span (MSS) (Roberts et al., 2020; Guu et al., 2020) are widely used tasks for pre-training. ICT uses '*input-passage*' pairs that have explicit supervision for training passage-wise attention, but has no supervision for the answer generation. On the other hand, MSS trains the model by '*input-output*' pairs that have strong supervision for the answer generation, but no supervision for the retriever, requiring additional warm-up training for retrieval such as ICT. Thus, these tasks are sequentially applied to pre-train the pipeline models to overcome their limitations (Guu et al., 2020; Singh et al., 2021). However, we train our single model architecture with retrieval, reranking, and reading layers at the same time using triples of '*input-passage-output*' for pre-training. To provide such supervisions, we extend a masked salient span task with explicit passage labels.

Masked Salient Span with Passage Labels (MSS-P): We first pick one named entity and mask all instances of this entity from the sentence. We explicitly add a ground truth passage that contains the masked named entity from 2 previous and next passages except its original passage. We refine the data by simple heuristics (using only pairs of sentence and target passage that contain at least 1 common named entity other than the masked span, and selecting the target passage with the highest number of common named entities when there are multiple passages containing the masked span). In this way, we generate 53M triples from Wikipedia passages in total.

3.3 Training Procedure

The model is pre-trained and fine-tuned iteratively to refresh retrieved passages for a better approximation of the distribution over all the passages.

We start the first pre-training iteration using the initial pre-training data, extracted by the MSS-P method that has one ground-truth passage for each query sentence. Note that with one positive passage per query, $\mathcal{L}_{retrieval}$ is equivalent to the negative log-likelihood loss of predicting the positive passage along with negative passages. However, $\mathcal{L}_{reranking}$ is 0 at this pre-training iteration and does not yield any training signal because it can only learn from contrasting multiple retrieved passages.

From the second iteration, the model is trained with 100 passages fetched from the retrieval layer. We add the original ground truth passage to the retrieved passages to ensure that the model learns to refer to the knowledge instead of implicitly memorizing the answer in its internal parameters. We do not filter more passages at the reranking layer during training to compute $score_{decoder}$ of Equation (5) to allow the reader to learn from the maximum number of passages. We pre-train the model for several iterations until the performance of the retrieval converges based on the recall metric.

After the pre-training, the model is then fine-tuned following the same procedure as that after the first iteration. To prevent an over-fitting of the reader due to the limited size of the fine-tuning data, we simply re-initialize the model with the pre-trained YONO model after retrieval performance converges following Izacard and Grave (2021a). Note that the model is fine-tuned by only weak-supervision of question-answer pairs without gold passages.

4 Experiments

4.1 Model Configurations

We primarily compare our model with baselines that use 440M parameters in total. To get a single language model with 440M parameters, we initialize our model from the pre-trained T5-large (Raffel et al., 2020) discarding final 18 decoder layers. This results in our model with 24 encoder and 6 decoder layers. The retrieval layer uses the first 12 encoder layers that uses 25% fewer parameters than baselines' bi-encoders retrievers (165M vs 220M). Since our reranking layer works on the representations of the retrieval layer, we only allocate 4 encoder layers for reranking. The total number of parameters used for retrieval and reranking is 220M. The remaining 220M parameters are allocated for the reading layer.

4.2 Training Details

At the first pre-training iteration, the model is trained with a batch of 800 *question-passage-answer* triples for 100K steps. From the second iteration, we train the model for 1,250 steps per iteration using a batch with 64 *question-passages-answer* triplets, where each triplet is packed with

Model	Passage Label	Aug. data	Model # Params	Natural Questions R@5	Natural Questions R@20	Natural Questions R@100	TriviaQA R@5	TriviaQA R@20	TriviaQA R@100
BM25 (Mao et al., 2021a)			-	43.6	62.9	78.1	67.7	77.3	83.9
DPR (Karpukhin et al., 2020)	√		220M	68.1	80.0	85.9	-	79.4	85.0
DPR^{new} (Karpukhin et al., 2020)	√		220M	72.2	81.3	87.3	-	-	-
GAR^{+} (Mao et al., 2021a)	√	√	220M	70.7	81.6	88.9	76.0	82.1	86.6
PAIR (Ren et al., 2021)	√	√	220M	74.9	84.0	89.1	-	-	-
coCondenser (Gao and Callan, 2021)	√		220M	**75.8**	84.3	89.0	76.8	83.2	87.3
DPR-PAQ (Oguz et al., 2021)	√	√	220M	74.2	84.0	89.2	-	-	-
ANCE (Xiong et al., 2021a)	√		220M	-	81.9	87.5	-	80.3	85.2
E2NR (Sachan et al., 2021)			220M	75.0	84.0	89.2	76.8	83.1	87.0
R2-D2$_{Retrieval}$ (Fajcik et al., 2021)	√		220M	68.6	80.6	86.7	69.8	78.9	84.7
FiD-KD (Izacard and Grave, 2021a)	√		220M	73.8	84.3	89.3	**77.0**	**83.6**	**87.7**
Larger models									
E2NR (Sachan et al., 2021)			660M	76.2	84.8	89.8	78.7	84.1	87.8
DPR-PAQ (Oguz et al., 2021)	√	√	660M	76.9	84.7	89.2	-	-	-
YONO$_{Retrieval}$			165M	75.3	**85.2**	**90.2**	76.8	83.5	87.4
Reranker models									
GAR^{+}-BART (Mao et al., 2021b)	√		406M	73.5	82.2	-	-	-	-
GAR^{+}-RIDER (Mao et al., 2021b)	√		110M	75.2	83.2	88.9	77.9	82.8	85.7
R2-D2$_{Reranking200}$ (Fajcik et al., 2021)	√		110M	76.8	84.5	88.0	78.9	83.5	86.0
YONO$_{Reranking200}$			55M	**79.1**	**86.7**	90.7	82.1	86.0	88.1
YONO$_{Reranking800}$			55M	**79.1**	86.6	**91.1**	**82.3**	**86.4**	**88.7**

Table 1: Recall@N results on Natural Questions and TriviaQA test sets. The best retrieval and reranking scores except larger models are indicated in bold. Reranking200/800 refer to reranking the 200/800 retrieved passages. The GAR^{+} model uses a further 406M params for augmenting the query.

100 retrieved passages. In total, we run 42 additional iterations after the first iteration for pre-training.

After pre-training, the model is fine-tuned the same way as pre-training, except it is trained for 1 epoch at every iteration.

The model is optimized with the Adam optimizer (Kingma and Ba, 2015) with a learning rate 10^{-4}. The first iteration takes 24 hours, and other iterations take around 5 hours each including MIPS indexing and passage refresh on 8 A100 GPUs. The penalty γ for attention scores of the random in-batch negative passages is set to 5 in all our experiments.

We found that answer generation more easily over-fits compared to the retrieval during fine-tuning. To prevent this over-fitting, the model is once reinitialized from the pre-trained YONO model at the 6^{th} iteration after the model achieves acceptable recall on the downstream task.

4.3 Datasets

We evaluate our model with two standard open-domain question answering datasets, Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017) following short answer subsets processed by Lee et al. (2019). Our external knowledge base is built using the Wikipedia dump from Dec. 20, 2018, where articles are split into passages of 100 words without overlap which is the same as datasets used in Karpukhin et al. (2020); Izacard and Grave (2021b); Singh et al. (2021) for a fair comparison.

5 Evaluation

5.1 Retrieval Performance

Table 1 and Table 2 show overall performance of our model and other baselines on Natural Questions and TriviaQA test sets. Our retrieval layer achieves the state-of-the-art recall@20/100 on Natural Questions regardless of model size even when compared with models with more than 4x model parameters. On TriviaQA, ours performs slightly worse than the state-of-the-art models, FiD-KD (Izacard and Grave, 2021a) and E2NR (Sachan et al., 2021), which use passage labels during training or 4x more parameters. Our approach achieves such performance without using passage labels making relevance for a larger range of applications that may not have these annotations. These results also do not use augmented data and as we show in Section 6.3 it only gives slight improvements on the end-to-end performance.

5.2 Reranking Performance

As shown in Table 1, the reranking layer further improves the recall of our retrieved passages by 3.8 and 5.5 absolute recall@5 on NQ and TQA respectively when reranking 800 retrieved passages. This

Model	# Params	NQ	TQA
Discriminative models			
OrQA (Lee et al., 2019)	330M	33.3	45.0
REALM (Guu et al., 2020)	330M	40.4	-
ANCE (Xiong et al., 2021a)	330M	46.0	57.5
Generative models			
RAG (Lewis et al., 2020)	440M	44.5	56.8
FiD (Izacard and Grave, 2021b)	440M	48.2	65.0
FiD-KD (Izacard and Grave, 2021a)	440M	49.6	68.8
E2NR (Sachan et al., 2021)	440M	45.9	56.3
EMDR^2 (Singh et al., 2021)	440M	52.5	71.4
Larger models			
E2NR (Sachan et al., 2021)	1.4B	48.1	59.6
FiD (Izacard and Grave, 2021b)	990M	51.4	67.6
FiD-KD (Izacard and Grave, 2021a)	990M	53.7	72.1
UnitedQA (Cheng et al., 2021)	2.09B	54.7	70.5
R2-D2 (Fajcik et al., 2021)	1.29B	55.9	69.9
YONO$_{Retrieval}$	440M	53.2	71.3
YONO$_{Reranking200}$	440M	53.2	71.5
YONO$_{Reranking800}$	440M	**53.2**	**72.1**

Table 2: End-to-end Open QA Exact-Match results on Natural Questions and TriviaQA test sets. Our model uses top 100 retrieved or reranked passages to generate answers. The best EM scores except larger models are indicated in bold.

is 2.3 and 3.4 absolute point improvements over the previous state-of-the-art reranker model. Our model achieves these recall performances using only 55M parameters which is only half the size of the other reranker models. Similar to our retriever, our reranker does not require passage labels, unlike other rerankers. This improvement in recall when using the reranker persists even when reranking only 200 passages.

5.3 End-to-end Performance

Our model achieves the best end-to-end performance among the models of the same size on NQ and irrespective of the model size on TQA as shown in Table 2. Our best scores improve EM scores by 0.7 points on both NQ and TQA respectively over the previously best performing model of the same size, EMDR^2 (Singh et al., 2021). Using data augmentation further boosts these improvements on NQ by 0.3 as shown in Table 5. Using reranking also improves the end-to-end scores on TQA by 0.8, a negligible improvement on NQ. We conjecture that this may be due to the higher recall of the retriever on NQ.

6 Ablation Studies

6.1 The Reader Loss on Retrieval Performance

The retrieval layer is trained by signals from both retrieval and reader losses. While the retrieval loss directly trains the retrieval scores, the reader loss is also a useful indirect training signal for the retriever. This signal is a key advantage of our approach over similar works such as Izacard and Grave (2021a); Singh et al. (2021). We evaluate the performance gain from the additional generation loss at the first pre-training iteration as shown in Table 3. The model trained with both losses shows significant improvement over the model trained with only the retrieval loss. These relative gains are larger the fewer the number of retrieved passages. This result shows that reader's generation loss is very effective for training the retriever. We evaluate after the first iteration because the reader loss is necessary for training with multiple retrieved passages in the following iterations.

Loss	R@5	R@20	R@100
$\mathcal{L}_{retrieval} + \mathcal{L}_{reader}$	28.8	48.1	67.0
$\mathcal{L}_{retrieval}$	18.0	32.1	49.7
Δ	+10.8 (60.0%)	+16.0 (49.8%)	+18.7 (34.8%)

Table 3: Effect of reader's generation loss on zero shot retrieval performance after the first iteration of pre-training on Natural Questions development set.

6.2 Shared Representations on Reader Performance

Model	Natural Questions	TriviaQA
YONO Reader	51.4	70.0
Stand-Alone Reader	48.0	67.8
Δ	+3.4 (7.1%)	+2.2 (3.2%)

Table 4: Effect of sharing of retrieval and reranking representations on exact match scores of reader models that use 220M parameters on NQ and TQA development sets.

Our reading layer uses 220M parameters but shares representations encoded by its preceding retrieval and reranking layers which use another 220M parameters. To measure gains of the shared representations, we compare our reader performance with that of a stand-alone reader model that uses 220M parameters that is the same as our reading layers. For a fair comparison, the stand-alone reader model is pre-trained and fine-tuned for the same amount of training tokens using the data retrieved by our YONO retriever. Table 4 shows that the reader model sharing representations outperforms the stand-alone reader by 7.1% and 3.2% on NQ and TQA respectively.

Pre-training	Natural Questions			TriviaQA		
	R@20	R@100	EM	R@20	R@100	EM
Retrieval						
No-pretrain	75.9	84.4	46.7	60.5	77.7	55.1
MSS-P(1 iter.)	83.4	89.0	51.5	80.4	85.9	68.4
MSS-P	85.2	90.2	53.2	**83.5**	87.4	**71.3**
MSS-P+ASGen	**85.5**	**90.3**	**53.5**	**83.5**	**87.5**	70.9
Reranking 200						
No-pretrain	81.4	85.9	46.8	75.8	79.7	57.4
MSS-P(1 iter.)	85.7	89.7	51.2	84.0	86.6	69.1
MSS-P	86.7	90.7	**53.2**	86.0	88.1	**71.5**
MSS-P+ASGen	**87.2**	**90.9**	**53.2**	**86.2**	**88.2**	71.2

Table 5: Effect of MSS-P pre-training and further pre-training using augmented data on Natural Questions and TriviaQA test sets.

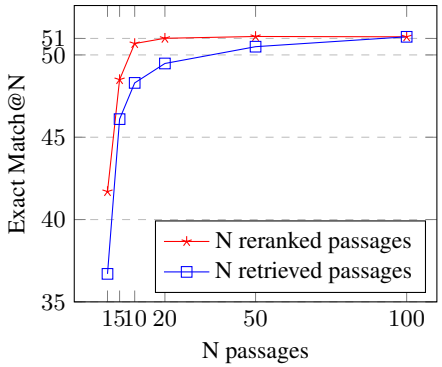

Figure 2: Exact Match scores for given N retrieved or reranked passages on NQ development set. Rerank EM scores are from reranking only 100 retrieved passages.

	Input Len.	First Stage Passages	First Stage Steps	Second Stage Passages	Second Stage Steps	Total Tokens
REALM	288	4096	100K	4096	200K	352B
EMDR^2	256	4096	100K	3200	82K	171B
YONO	200	800	100K	6400	52.5K	**83B**

Table 6: Total pre-training tokens. The first and second stages of REALM and EMDR^2 are ICT and MSS pre-training respectively.

6.3 Effectiveness of MSS-P pre-training

To show the effectiveness of our MSS-P pre-training method, we evaluate this by fine-tuning our architecture without any pre-training using initial retrievals from DPR (Karpukhin et al., 2020) and fine-tuning after the first pre-training iteration. We also compare our pre-training to that of the additional data augmentation (Mao et al., 2021a; Ren et al., 2021; Oguz et al., 2021). We generate 'question-answer' pairs from a Wikipedia dump using a question and answer generation model trained on the NQ dataset using the ASGen approach (Back et al., 2021). The model is further trained after the pre-training by this augmented data for 12 more iterations before fine-tuning.

Table 5 shows retrieval, reranking, and reading performance on Natural Questions and TriviaQA test sets. Our MSS-P pre-training dramatically boosts the performance of our architecture by 4.8 and 13.3 EM points on NQ and TQA even with only the first pre-training iteration. Further iterations of our pre-training improve EM by 1.7 and 2.9 EM points. The further data augmentation pre-training improves performance on NQ consistently but only slightly, while the improvements on TQA are inconsistent, as the data was generated by the model trained on NQ dataset. These results clearly demonstrate that our simple self-supervised MSS-P pre-training is strong enough to compete favorably against sophisticated data augmentation approaches.

7 Analysis

7.1 Computational Efficiency of Reranking

In many dense retrieval systems, a reranker is often omitted due to functional overlaps with the reader and computational overhead (Guu et al., 2020; Lewis et al., 2020; Singh et al., 2021). Thanks to the shared representations across the reader and reranker, our model can incorporate a reranking function without significantly more parameters or computation. By dropping irrelevant passages early at the reranking layer, we can achieve better computational efficiency. Figure 2 shows exact match scores for given N retrieved or reranked passages. The model still achieves optimal EM performance with only the top 20 reranked passages reranked from 100 retrieved passages. Reranking 100 to 20 passages can reduce the inference computation by 27.4% without pre-computed passage representations, and 54.0% with pre-computed passage representations without losing end-to-end performance.

7.2 Pre-training Efficiency

The number of pre-training tokens is an important metric to measure the efficiency of the pre-training objective. As shown in Table 6, REALM (Guu et al., 2020) and EMDR^2 (Singh et al., 2021) use 352B and 171B tokens in total respectively. In contrast, our method uses only 83B tokens, which is 76.4% and 51.5% less than the training tokens used to train REALM and EMDR^2 respectively. Furthermore, the retrieval index is updated only 43 times during our pre-training, while EMDR^2 updates the index 164 times. This is a significant reduction of

the computation overhead for pre-training.

8 Related Works

Neural Retriever Augmented Language Modeling (NRALM): Augmenting language models with neural retrieval has been shown to be very effective, such as by retrieving nearest neighbor words for LM tasks (Khandelwal et al., 2020; Yogatama et al., 2021) or Machine Translation (Khandelwal et al., 2021). Dinan et al. (2019) proposed a decomposed transformer for conversation tasks, which enabled pre-computation of the external knowledge embeddings.

ORQA (Lee et al., 2019) proposed the ICT task to pre-train a decomposed retriever, and DPR (Karpukhin et al., 2020) enhanced this approach with in-batch negatives and hard negatives to eliminate the pre-training. Synthetic Data Augmentation is also commonly used, such as in DPR-PAQ (Oguz et al., 2021), PAIR (Ren et al., 2021), Hu et al. (2021). Per-token embeddings or multiple embeddings were used in ColBERT (Khattab et al., 2020), ME-BERT (Luan et al., 2021), Lin et al. (2021), Lee et al. (2021).

Similar to our approach of re-ranker on top of a shared retriever, PreTTR (MacAvaney et al., 2020) pre-computed term representations for all documents, and used these to run only the upper layers of a transformer reranker model. Decoupled Transformer (Elfdaeel and Peshterliev, 2021) also shares the lower layers of a transformer encoder to serve as a reranker, using the upper layers as a reader and focuses on computationally efficient reranking. Our approach extends these approaches by also incorporating a retriever and a decoder in the model.

E2E Optimization of NRALM: It is intractable to re-compute the embeddings of the knowledge for every weight update. REALM (Guu et al., 2020) and ANCE (Xiong et al., 2021a) proposed async index refresh to propagate updates to the index to yield better negatives. TAS (Hofstätter et al., 2021) and Xiong et al. (2021b) used clustering of embeddings for the same. RAG (Lewis et al., 2020) used DPR with BART generator to marginalize over generated tokens, which is back-propagated to the retriever. REALM++ (Balachandran et al., 2021) added a re-ranker to REALM.

Similar to our work, Bruyn et al. (2020) and TREAD (Shuster et al., 2021) utilize BART and T5 reader's encoders as a retriever. In contrast to these methods, our work has a unified pre-training method to train all the components of the model. Furthermore, our model also has an integrated re-ranker, and the query and passage are cross-encoded for more expressive representations.

Multi-passage Readers: Reading multiple passages at the same time is difficult, as concatenating multiple passages increases computation quadratically for transformers. Zhao et al. (2020) reduced multiple passages and sentences to few via a knowledge selector, which were then concatenated and passed on to GPT (Radford et al., 2019). FiD (Izacard and Grave, 2021b) concatenated the encoded representations of documents, which can then be attended by the decoder, achieving large performance gains. This approach was also applied in RocketQA (Qu et al., 2021). UnitedQA (Cheng et al., 2021) and R2D2 (Fajcik et al., 2021) combine results from an ensemble of extractive and generative readers, whereas PAQ (Lewis et al., 2021) directly retrieves answers with an FiD fallback.

Similar to our work, both REALM (Izacard and Grave, 2021a) and EMDR^2 (Singh et al., 2021) train the retriever with a signal from the reader. Unlike these approaches, our model has shared lower layers for more effective utilization of model parameters and better end-to-end gradient flow across the whole model. Furthermore, our training methodology results in propagating the answer generation loss of the retriever, which has a large effect on performance as we show in Table 3.

9 Conclusion

In this paper, we propose a novel language model architecture that embeds the retriever and the reranker as internal passage-wise attention mechanisms and a training method to effectively train this model. This singular model architecture efficiently uses model capacity by cascading and sharing the representations from retriever to reranker to the reader leading to better gradient flow for end-to-end training. We evaluate our model on Natural Questions and TriviaQA open datasets and for a fixed parameter budget, our model outperforms the previous state-of-the-art model by 1.0 and 0.7 exact match scores. We show detailed ablations and analyses of each component of our approach. Our future work is to conduct more experiments on various knowledge-intensive tasks and extend this model to match query and passage in multiple or hierarchical representation spaces.

Limitations

One caveat of sharing representation for multiple tasks like retrieval, reranking, and reading is that these show different over-fitting tendencies during fine-tuning where the training data is limited. We found that answer generation over-fits more easily compared to the retrieval. Answer generation relies on more expressive representation via cross attention, which may make it easier to memorize the output and hence make it more vulnerable to over-fitting. Furthermore, at the first fine-tuning iteration, the model is trained by zero-shot retrieval results from the pre-trained model that has a relatively low recall rate and can harm the answer generation training. To refresh the over-fitted answer generation parameters, and to start from training data with a high recall rate, we simply re-initialize the model with the pre-trained YONO model after a few fine-tuning iterations. However, we believe that this issue should be addressed carefully using a more sophisticated solution. We further discuss the over-fitting issue and effect of re-initialization in Appendix A.

References

Seohyun Back, Akhil Kedia, Sai Chetan Chinthakindi, Haejun Lee, and Jaegul Choo. 2021. Learning to generate questions by learning to recover answer-containing sentences. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1516–1529, Online. Association for Computational Linguistics.

Vidhisha Balachandran, Ashish Vaswani, Yulia Tsvetkov, and Niki Parmar. 2021. Simple and efficient ways to improve REALM. *CoRR*, abs/2104.08710.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Maxime De Bruyn, Ehsan Lotfi, Jeska Buhmann, and Walter Daelemans. 2020. BART for knowledge grounded conversations. In *Proceedings of the KDD 2020 Workshop on Conversational Systems Towards Mainstream Adoption co-located with the 26TH ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD 2020), Virtual Workshop, August 24, 2020*, volume 2666 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021. UnitedQA: A hybrid approach for open domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3080–3090, Online. Association for Computational Linguistics.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Haytham Elfdaeel and Stanislav Peshterliev. 2021. Decoupled transformer for scalable inference in open-domain question answering. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), Held Online, 1-3September, 2021*, pages 386–393. INCOMA Ltd.

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. 2021. R2-D2: A modular baseline for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 854–870. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *SIGIR '21: The 44th*

International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, pages 113–122. ACM.

Ziniu Hu, Yizhou Sun, and Kai-Wei Chang. 2021. Relation-guided pre-training for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3431–3448. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021a. Distilling knowledge from reader to retriever for question answering. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Gautier Izacard and Edouard Grave. 2021b. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7:535–547.

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.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2020. Relevance-guided supervision for openqa with colbert. *CoRR*, abs/2007.00814.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

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.

Jinhyuk Lee, Alexander Wettig, and Danqi Chen. 2021. Phrase retrieval learns passage retrieval, too. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3661–3672. Association for Computational Linguistics.

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.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Patrick S. H. Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *CoRR*, abs/2102.07033.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online. Association for Computational Linguistics.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and atten-

tional representations for text retrieval. *Trans. Assoc. Comput. Linguistics*, 9:329–345.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 49–58. ACM.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021a. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100, Online. Association for Computational Linguistics.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021b. Reader-guided passage reranking for open-domain question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 344–350, Online. Association for Computational Linguistics.

Barlas Oguz, Kushal Lakhotia, Anchit Gupta, Patrick S. H. Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen-tau Yih, Sonal Gupta, and Yashar Mehdad. 2021. Domain-matched pre-training tasks for dense retrieval. *CoRR*, abs/2107.13602.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

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

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei 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.

Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, Online. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Devendra Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. 2021. End-to-end training of neural retrievers for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6648–6662, Online. Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3784–3803. Association for Computational Linguistics.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018. Evidence aggregation for answer re-ranking in open-domain question answering. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021a. Approximate nearest

neighbor negative contrastive learning for dense text retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Wenhan Xiong, Hong Wang, and William Yang Wang. 2021b. Progressively pretrained dense corpus index for open-domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2803–2815, Online. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Trans. Assoc. Comput. Linguistics*, 9:362–373.

Xueliang Zhao, Wei Wu, Can Xu, Chongyang Tao, Dongyan Zhao, and Rui Yan. 2020. Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390, Online. Association for Computational Linguistics.

Appendix

A Model Re-initialization during Fine-tuning

To overcome over-fitted reader parameters, we refresh the model parameter using the pre-trained YONO model at the fine-tuning iteration where the EM score starts to drop.

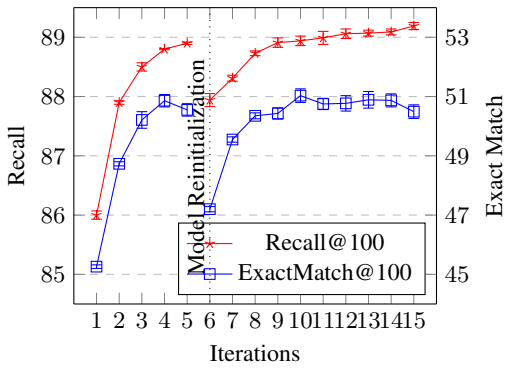

Figure 3: Average Recall and EM scores at each fine-tuning iteration with standard error bars from 3 runs on NQ development set. The model is once reinitialized at the 6^{th} iteration.

Figure 3 shows retrieval and end-to-end performance at each fine-tuning iteration. The Exact Match score drops at the 5^{th} iteration while the retrieval score keeps increasing. After re-initializing the model before the 6^{th} iteration, the model restarts with a higher recall and EM score. However, the EM score drops again from the 10^{th} iteration after achieving the best end-to-end performance, while the retrieval performance continues to improve. We leave further approaches for preventing over-fitting of our model such as freezing the model partially as future work.

B Effect of Pre-training Iterations on Retrieval Performance

Figure 4 shows recall@N at each training stage across the pre-training and fine-tuning using Natural Question development set. The first iteration of the pre-training results in zero-shot recall@100 of 67.0%, which is further improved by additional pre-training iterations to 71.8% recall@100. These zero-shot recall scores enable us to fine-tune our model without passage labels resulting in state-of-the-art retrieval and reranking performance.

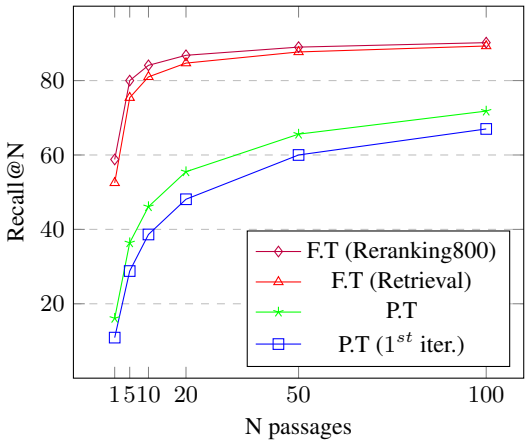

Figure 4: Recall@N at each training stage on NQ development set. P.T denotes pre-training, F.T denotes fine-tuning.

C Experiment Details

On Table 7, 8, and 9, we provide all training details and parameters used to conduct experiments on this paper.

D Raw Values for Plots in Figures

In Table 10 and 11, we provide raw values for plots in Figure 2 and 4.

Parameters	Values	
Initial model	T5-Large	
Dimensions		
- Model	1,024	
- Feed Forward	4,096	
- Attention head	64	
Attention head count	16	
# of layers and parameters		
- Total	24 enc + 6 dec	440M
- Retrieval layer	12 enc	165M
- Reranking layer	4 enc	55M
- Reading layer	8 enc + 6 dec	220M

Table 7: Model Parameters.

Parameters	Values
Pre-training	
Total iterations	43
Training tokens	
- Total	83B
- The 1^{st} iteration	16B
- One iteration from 2^{nd}	1.6B
- One batch	160K
Fine-tuning	
Best scoring iteration	NQ 10 / TQA 11
Training tokens	
- Total to the best iteration	NQ 15.8B / TQA 17.3B
- One iteration	NQ 1.58B / TQA 1.58B
- One batch	160K
Optimization	
Learning rate	10^{-4} (fixed)
Drop-out	0.1
Precision	float32
Gradient clipping	1.0
Gradient accumulation	
- The 1^{st} pretraining iteration	None
- otherwise	8 batches

Table 8: Training Parameters.

Passages	Retrived EM	Reranked EM
1	36.7	41.7
5	46.1	48.5
10	48.3	50.7
20	49.5	51.0
50	50.5	51.1
100	51.1	51.1

Table 10: Raw Values for EM for Figure 2

Passages	FT Rerank_{800}	FT Retrieval	PT	PT 1 iter.
1	58.8	52.5	16.1	10.9
5	80.0	75.4	36.4	28.8
10	84.1	81.0	46.1	38.6
20	86.8	84.7	55.5	48.1
50	89.0	87.7	65.6	60.0
100	90.2	89.3	71.8	67.0

Table 11: Raw Values for Recall@N for Figure 4

E Measures of Central Tendencies for Results

To measure the sensitivity of our model to varying seeds, we run 3 fine-tunings of our model on NQ, and report the mean and standard errors on the development set below, as shown in Figure 3. The model training seems stable with little variation across runs. We did not run multiple instances of pre-training as it is computationally expensive.

Parameters	Values
Text Inputs	
Max question length	40
Max sequence length	200
Retrieval Index	
Dimension	1,024
Precision	float32
Index Size (21M passages)	81GB

Table 9: Other Parameters.

Finetuning Iter.	Exact Match@100	Recall@100
1	45.3 ± 0.1	86.0 ± 0.1
2	48.7 ± 0.1	87.9 ± 0.0
3	50.2 ± 0.3	88.5 ± 0.1
4	50.9 ± 0.2	88.8 ± 0.1
5	50.6 ± 0.2	88.9 ± 0.1
6	47.2 ± 0.1	87.9 ± 0.1
7	49.6 ± 0.1	88.3 ± 0.1
8	50.4 ± 0.1	88.7 ± 0.1
9	50.4 ± 0.2	88.9 ± 0.1
10	51.0 ± 0.2	88.9 ± 0.1
11	50.8 ± 0.2	89.0 ± 0.1
12	50.8 ± 0.3	89.1 ± 0.1
13	50.9 ± 0.3	89.1 ± 0.1
14	50.9 ± 0.2	89.1 ± 0.1
15	50.5 ± 0.2	89.2 ± 0.1

Table 12: Raw values for Central Tendency and Standard Error on NQ development set for 3 finetuning runs of our model, as shown in Figure 3

F Links to Source Code and Datasets

The source code is based on the original implementation of FiD (Izacard and Grave, 2021b), which can be found at their Github.

Data for the Wikipedia dump, Natural Questions, and TriviaQA can also be downloaded from FiD's github using this script.

G Evaluation Metrics and Scripts

The evaluation script is based on the original FiD script.

Exact Match - This is the average across all examples of the per-example exact match score, which is 0 or 1 if all the words in the generated answer exactly match the annotated answer after unicode normalization by lower-casing, removing punctuation and spaces.

Recall@N - Recall@N measures the percentage of examples for which atleast one the top-N pas-

sages contains a span that matches the annotated answer as in Exact Match above.

H Dataset Statistics

Table 13 provides the statistics of our evaluation datasets.

Dataset	# Train	# Dev	# Test
Natural Questions	79K	8.8K	3.6K
TriviaQA	79K	8.8K	11K

Table 13: Dataset Statistics

I Computing Infrastructure

GPU model - 8x Nvidia A100 80 GB. CPU Model - 2x AMD EPYC 7543 32-Core Processor. RAM - 1000GB. PyTroch version - 1.8.0+cu111. Huggingface Transformers version - 3.0.2.