Is Retriever Merely an Approximator of Reader?

Sohee Yang and **Minjoon Seo**
KAIST AI*
{sohee.yang,minjoon}@kaist.ac.kr

Abstract

The state of the art in open-domain question answering (QA) relies on an efficient retriever that drastically reduces the search space for the expensive reader. A rather overlooked question in the community is the relationship between the retriever and the reader, and in particular, if the whole purpose of the retriever is just a fast approximation for the reader. Our empirical evidence indicates that the answer is *no*, and that the reader and the retriever are complementary to each other even in terms of accuracy only. We make a careful conjecture that the architectural constraint of the retriever, which has been originally intended for enabling approximate search, seems to also make the model more robust in large-scale search. We then propose to distill the reader into the retriever so that the retriever absorbs the strength of the reader while keeping its own benefit. Experimental results show that our method can enhance the document recall rate as well as the end-to-end QA accuracy of off-the-shelf retrievers in open-domain QA tasks.

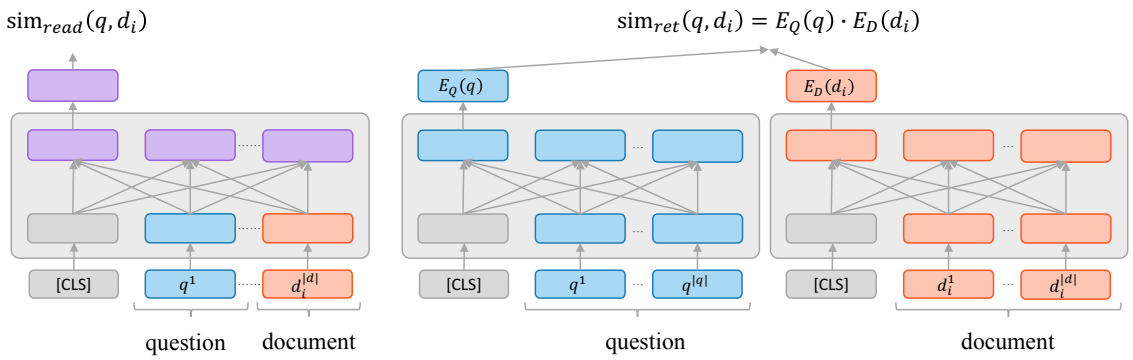
1 Introduction

The task of open-domain question answering can be defined as creating a model that takes a question and the knowledge source as the input and outputs the answer to the question. In this paper, we primarily focus on unstructured (text) knowledge data such as Wikipedia, and we do not consider structured sources such as Knowledge Graph. In most cases (Karpukhin et al., 2020; Lewis et al., 2020; Izacard and Grave, 2020), since the (unstructured) knowledge data is so big, one first *retrieves* a few relevant documents to the question from the knowledge data and then *reads* the retrieved documents to obtain the answer. For the retriever to quickly search over a large number of documents, its architecture is often constrained to be a *two-tower* (Figure 1b), where the question and the documents are independently mapped to a common vector space. This way, fast sublinear-time approximation methods such as approximate nearest neighbor search (Shrivastava and Li, 2014) can be utilized. The reader, on the other hand, leverages the freedom of the *one-tower* architecture (Figure 1a), which takes both the question and the document as a concatenated input and is allowed to process them jointly to obtain a more accurate answer. The reader, however, clearly has a linear-time complexity with respect to the input size.

It is hence commonly conceived that the main role of the retriever is the gain of efficiency at the cost of accuracy. Theoretically, this makes sense; the two-tower architecture enforces the information in the question or the document to be bottlenecked by their embeddings, which can cause the loss of information, and furthermore, they only interact through similarity (metric or inner product) space, which further limits its capability. This especially corresponds well with the motivation for the kernel method in SVMs (Cortes and Vapnik, 1995), where one would need an infinite number of dimensions for the feature map (two-tower) to exactly mimic even a simple kernel (one-tower) such as an RBF kernel (Chang et al., 2010) in inner product space. After all, any target function that a two-tower model can learn is clearly also learnable by a one-tower model.

Nevertheless, we find some empirical hints from the previous works that somewhat contradict this general belief. For instance, Clark and Gardner (2017) and Lewis et al. (2020) both demonstrate in Figure 3 that as the reader reads more top-k documents, the end-to-end QA accuracy somewhat decays. While these observations were not seriously discussed previously, they are clearly worth a closer look because they imply that their retrievers are not just reducing the search space for efficiency,

*The work was done while the authors were working at NAVER Corp.



(a) One-Tower Model (b) Two-Tower Model

Figure 1: Comparison between the architectures of a one-tower model, e.g., reader, and a two-tower model, e.g., retriever, with BERT (Devlin et al., 2019). The one-tower model takes the question and document together and jointly processes them throughout all the layers, while the two-tower architecture separately models the question and document whose outputs interact only at the inner product space of the final embeddings.

but also playing some role in making the reader more accurate. Is it simply because the readers were just not trained properly?

In this paper, we delve into these observations and find that that the retriever is not merely an approximator of the reader for the document ranking purpose. In the first part (Section 3), we empirically show that the two-tower model (retrieval-based approach) is not only efficient but also essential for creating a good open-domain question answering model. That is, the retriever and the reader are complementary to each other, where each has a comparative advantage over the other for accuracy. We guess that the architectural constraint of the retriever, which might have been originally intended for an approximation, seems to also make the model more robust against various negative examples in large-scale search. Then in the second part (Section 4), we propose a distillation method to enhance the retriever so that the retriever can absorb the strength of the reader while keeping its comparative benefit. Our method is able to enhance the recall rate and the end-to-end QA accuracy of off-the-shelf retrievers (Karpukhin et al., 2020), especially with a significant improvement in top-1 recall and accuracy.

2 Related Work

Open-Domain QA Answering open-domain factoid questions can utilize either a highly structured knowledge source (e.g., Knowledge Graph) or large unstructured data (e.g., Wikipedia). In this paper, we primarily focus on the latter, and more specifically "Retrieve & Read" paradigm (Chen et al., 2017; Guu et al., 2020), which has been steadily predominant in the community and yielding both high accuracy and efficiency. Note that the act of "reading" can be either extractive (i.e., the finding where the answer phrase lies in the documents) (Karpukhin et al., 2020) or generative (Wang and Jiang, 2017; Lewis et al., 2020; Izacard and Grave, 2020). More recently, "closed-book" QA models have been drawing attention (Roberts et al., 2020; Brown et al., 2020), where all knowledge is encoded in the parameters of the model, and no external knowledge source is attached (i.e., fully parametric). These models are however also known for being computationally expensive since they require many orders of more parameters to *memorize* the knowledge corpus, and the answers are often unreliable and uninterpretable. Another line of recent work (Seo et al., 2019) makes the task as a pure retrieval problem by indexing all answer phrase candidates in the knowledge corpus, which makes the inference more efficient and end-to-end. Nevertheless, these models can only yield only extractive answers (than generative) and require much larger storage to keep the index.

Document Retrieval & Ranking Retrieving relevant documents given a query has been a significant interest in both research and industry, due to its direct applicability to search engines and question answering systems. It has been known in the community that directly providing the retrieved documents in the order of their retrieval scores often results in poor search quality (Lee et al., 1997). Therefore, in practice, most search engines utilize a ranker that looks into the retrieved top-k documents and reorders them (Carbonell and Goldstein, 1998;

Nogueira and Cho, 2019). Unlike the retriever, the ranker often employs the one-tower architecture to fully prioritize the search quality over efficiency. From the QA perspective, ranking can be considered to be either explicitly (Karpukhin et al., 2020) or implicitly (Lewis et al., 2020) embodied in its reader component.

Similarity Search In order to find a relevant item (e.g., document) in an extremely large search space, the items and the query are often embedded into a common vector space (using two-tower architecture), and then an off-the-shelf similarity search library such as `faiss` (Johnson et al., 2019)[1] can be used for fast retrieval. The search process can be especially more efficient when an approximation is used. Traditionally, metric space such as cosine distance or L2 has been a popular choice for the similarity function, which allows us to use approximation methods such as Locality Sensitive Hashing (LSH) (Gionis et al., 1999) or k-means clustering (Hartigan and Wong, 1979) by leveraging the nice theoretical properties of metric space. However, for many recent question answering models, inner product space seems to lead to a similar or better model than L2 during training (Seo et al., 2019; Karpukhin et al., 2020), where one can still utilize asymmetric LSH (aLSH) (Shrivastava and Li, 2014) or k-means clustering. We also adopt inner product space with k-means clustering.

3 Is Retriever Merely an Approximator of Reader?

In Section 3.1, we first formally define the open-domain question answering task, in order to formulate our question of interest in a formal manner as well. Then in Section 3.2 and 3.3, we provide several empirical explanations for the relationship between the retriever and the reader.

3.1 Problem Formulation

The task of open-domain QA is to learn a *reader* function $\boldsymbol{f}$ that maps two inputs, question $\boldsymbol{q}$, and knowledge data $\boldsymbol{d}$, to the answer $\hat{\boldsymbol{a}}$, i.e., $\hat{\boldsymbol{a}} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{d})$. Note that in the case of closed-book question answering (as discussed in Section 2), the knowledge corpus is only observed during training, so the function only takes the question as the input. Here, we will focus on the more typical open-book case, where the identical knowledge corpus is observed during both training and inference.

[1] https://github.com/facebookresearch/faiss

The classic challenge of the task is that the knowledge corpus $\boldsymbol{d}$ is too big, so applying the linear-time function $\boldsymbol{f}$ on the entire $\boldsymbol{d}$ is intractable. Therefore, a common practice is to create a *retriever* function $\boldsymbol{g}$, which extracts a subset of the knowledge corpus in sublinear-time, that would be small enough for efficiency and has sufficient information to infer the answer to the question. The retriever usually adopts two-tower architecture (Figure 1b), whose structural constraint allows us to efficiently perform the subset extraction via approximate maximum inner product search (or nearest neighbor search). More concretely, the subset $\boldsymbol{d}' \subset \boldsymbol{d}$ (assuming $\boldsymbol{d}$ consists of N chunks such as documents, i.e., $\boldsymbol{d} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N\}$ for convenience) is obtained as:

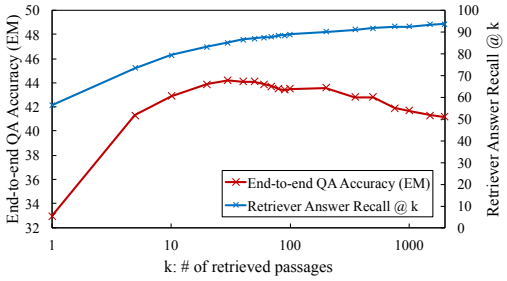
$$\boldsymbol{d}' = \boldsymbol{g}(\boldsymbol{q}, \boldsymbol{d}) = \text{top-k}_{\boldsymbol{d}_i} \psi(\boldsymbol{q}) \cdot \phi(\boldsymbol{d}_i),$$

where k is the target number of chunks, ψ, ϕ are feature map functions that map the question and each knowledge chunk to a common d-dimensional vector space, respectively, and $\cdot$ is an inner product operator (we can replace it with L2-distance if we are using nearest neighbor search).

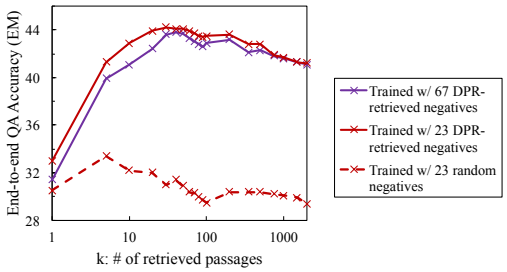
We let $\boldsymbol{f}'$ represent the resulting combined model, i.e., $\hat{\boldsymbol{a}} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{d}) \approx \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{g}(\boldsymbol{q}, \boldsymbol{d})) = \boldsymbol{f}'(\boldsymbol{q}, \boldsymbol{d})$. Then we can consider $\boldsymbol{f}'$ as an approximator of $\boldsymbol{f}$, which gives us efficiency benefit, possibly at the cost of accuracy. Indeed, an important limitation of $\boldsymbol{f}'$ is that the introduction of $\boldsymbol{g}$ is a strong structural constraint, which means it is not able to model some relationships that $\boldsymbol{f}$ can easily do. An easy example is an RBF Kernel (Chang et al., 2010), which can be easily modeled using a one-tower model but cannot be modeled in inner product space since it requires an infinite number of dimensions. Apparently, $\boldsymbol{f}$ is strictly more expressible than $\boldsymbol{f}'$, so the theoretical answer to whether $\boldsymbol{f}'$ has any accuracy (not efficiency) benefit compared to $\boldsymbol{f}$ would be no, and we would conclude that $\boldsymbol{f}'$ is *merely an approximator*. [2]

However, in practice, the structural constraint in $\boldsymbol{g}$ induces a representational bottleneck that seems to benefit $\boldsymbol{f}'$ accuracy-wise during training. That is, we think $\boldsymbol{f}'$ is *not merely an approximator* of $\boldsymbol{f}$. We will carefully explore this claim in this section. In Section 3.2, we first perform a case study on a popular reader and retriever model in the literature (Karpukhin et al., 2020) and observe that the

[2] That is, to be precise, we are comparing between the retriever-augmented reader and the pure reader.



(a) End-to-end QA accuracy (Exact Match, y-axis on the left) of DPR reader and the retrieval recall rate (y-axis on the right) of DPR retriever.



(b) End-to-end QA Accuracy (Exact Match) of DPR readers trained under various training environments.

Figure 2: Results of the experiments using DPR on 1,000 questions sampled from the test set of NaturalQuestions.

results agree with our hypothesis. We however note that there could be possibly other factors that may have affected this observation, so we adjust the training environment for a fair comparison in Section 3.3, and again observe the empirical evidence for our hypothesis.

3.2 Case Study on DPR

We first start with a simple case study on the off-the-shelf retriever and reader (DPR) by Karpukhin et al. (2020). An ideal, full-scale experiment would be to compare the reader's accuracy with and without the retriever. However, reading the entire knowledge corpus (e.g., Wikipedia) is infeasible given limited computational resources. We thus indirectly compare them on 1,000 questions randomly sampled from the test set of NaturalQuestions (NQ) (Kwiatkowski et al., 2019), by varying the number of retrieved documents for each question (k) from 1 to 2,000 (large but much more manageable than the total number of documents) and analyze the reader's accuracy with the retrieved documents. When $k = 1$, the retriever's influence on the reader's accuracy is maximized, as the correct document must be retrieved for the reader to get the right answer. As k increases, the retriever's influence decreases, and the reader gets more room for improvement at the cost of slower speed.

The red graph in Figure 2a shows the accuracy of the reader at $k = 1, \cdots, 2000$ (in log scale). We observe that its accuracy peaks at $k = 30$ and steadily decreases. This agrees with the previous observations by Clark and Gardner (2017) and Lewis et al. (2020) that the reader accuracy peaks at a relatively low value of k. While not seriously discussed before, this is a rather extraordinary behavior because the reader's architecture (one-tower, Figure 1a) is inherently more expressive than the retriever's architecture (two-tower, Figure 1b). One possible explanation for this discrepancy could be due to the different training setup between the retriever and the reader, where the retriever observes more negative examples (using in-batch negative training), which are also more random (top-k from random). Therefore, we do experiments below on DPR reader under a training environment similar to that of the retriever.

3.3 Reader Training under Setting Similar to Retriever's

Here we investigate whether the performance drop of the reader with increasing k after the peak comes from (1) training with less random negatives or (2) training with smaller number of negative examples. For the former investigation, we set the number of negatives examples to 23 and examine two sampling strategies; one is where the negative examples are randomly sampled from the entire corpus, and the other is where they are obtained from the top-k documents retrieved by the original DPR retriever. It is hence expected that the former training setup allows the reader to observe more diverse negative examples, while the latter setup allows the reader to observe harder negative examples, as also noted by Karpukhin et al. (2020). For the latter examination, we fix the sampling strategy as obtaining the negative examples from DPR retriever, and train the reader with different numbers of negative examples: 23 (original DPR reader) and 67. To train the reader with 67 negatives while preventing GPU OOM, we use the batch size of 8 and the gradient accumulation step of 2 (the original DPR reader uses the batch size of 16 for training).[3] We vary k

[3] By nature, the one-tower reader, which jointly models a question-document pair each time, is difficult to be trained with a large batch size or a large number of negative passages, unlike the two-tower retriever.

in the range of 1 to 2,000 and report the reader's accuracy in a similar manner as in Figure 2a.

Figure 2b shows the results of both experiments. From the red graphs, which represent the result of the former investigation, we can clearly see that the reader's accuracy decreases as k increases regardless of the sampling strategy, that the solid line (DPR-retrieved negatives) and dashed line (random negatives) peak at small k of 30 and 5, respectively. Likewise, increasing the number of negatives also does not seem to increase the number k where the peak is achieved; the peak for each of the models trained with 23 and 67 negatives are at similar k of 30 and 40, respectively. Such results that the reader's accuracy still peaks at a relatively low k even under the training setup similar to that of the retriever seem to indicate that the retriever is indeed giving a non-trivial accuracy benefit to the reader.

4 Distilling Reader into Retriever

As seen in Section 3, the two-tower retriever and one-tower reader are in a complementary relationship to each other; our guess is that the former is good at finding passage embeddings relevant to a question embedding from a large vector space, and the latter is good at distinguishing between difficult examples in a relatively small search space. Therefore, here we propose a method to combine the best of both worlds to enhance the performance of the retriever: *distilling the knowledge of the reader into the retriever*.

Several learning methods can be applied or explored to transfer the knowledge of the reader to the retriever, but we specifically utilize knowledge distillation (Hinton et al., 2014) in this work. Knowledge distillation is often used for model compression, enabling the generalization ability of a large model to be transferred to a smaller model without much loss of performance (Sanh et al., 2019; Liu et al., 2019). On the other hand, here we employ knowledge distillation to transfer the representational knowledge learned by the expressive one-tower reader to the two-tower retriever, which has a relatively bottlenecked architecture.

Hereafter, we describe how the two-tower retriever of DPR (Karpukhin et al., 2020), which is actively adopted in the latest open-domain question answering models to achieve state-of-the-art results (Lewis et al., 2020; Izacard and Grave, 2020), can be enhanced by distillation from the one-tower reader proposed in the same work. Although this paper presents one example of utilizing distillation, the approach itself is simple and intended to be generic, not strictly tied to any particular retriever or reader, as long as the knowledge of the reader can be distilled into the retriever in any form.

4.1 Distillation Setup

Student: Retriever Following the architecture of DPR retriever, our proposed READER-DISTILLED RETRIEVER (RDR) is a two-tower model that consists of two dense encoders: $E_Q(\cdot)$, which maps a question q to a vector space, and $E_D(\cdot)$, which maps each knowledge chunk d_i, a block of 100 words dubbed passage in this work, to a common vector space.

Let D_{ret} denote the list of passages that the retriever is scoring on behalf of a question q. Then, the retrieval score $\text{sim}_{\text{ret}}(q, d_i)$ is calculated as the inner product between q and each passage $d_i \in D_{\text{ret}}$. The top-scoring passages that serve as the input to the reader, D_{read}, are inferred as follows:

$$\text{sim}_{\text{ret}}(q, d_i) = E_Q(q) \cdot E_D(d_i),$$
$$D_{\text{read}} = \left\{ d_k \mid k \in \underset{i}{\text{argmax}}\ \text{sim}(q, d_i), \forall d_i \in D_{\text{ret}} \right\}.$$

Teacher: Reader DPR reader described in the work of Karpukhin et al. (2020) is the teacher of our retriever. As input, the model takes the concatenation of the token embeddings of question q, the title, and contents of each passage d_i in D_{read}. Along with the scores of each token being the starting or ending positions of the answer, the reader outputs a ranking score $\text{sim}_{\text{read}}(q, d_i)$ for each passage $d_i \in D_{\text{read}}$.

Objective Function Let each of z_{ret} and z_{read} be a $|D_{\text{read}}|$-dimensional vector, where each element is the score for a question-passage pair in the inputs to the reader, $(q, d_i), \forall d_i \in D_{\text{read}}$. For model $m \in \{\text{ret}, \text{read}\}$,

$$z_m = \left[\text{sim}_m(q, d_1), \ldots, \text{sim}_m(q, d_{|D_{\text{read}}|}) \right].$$

To turn the score vectors z_{ret} and z_{read} into probability distributions without saturation, softmax with temperature T is applied. We then minimize $D_{KL}(P_{\text{read}} || P_{\text{ret}})$, where each of the probability distributions is calculated as follows:

$$P_m = \left[\frac{\exp(z_{m_1}/T)}{\sum_j \exp(z_{m_j}/T)}, \cdots, \frac{\exp(z_{m_{|D_{\text{read}}|}}/T)}{\sum_j \exp(z_{m_j}/T)} \right].$$

Table 1: Retrieval recall rate of DPR (Karpukhin et al., 2020), RAG (Lewis et al., 2020), and RDR (Ours) on NQ-dev, NQ-test, and TriviaQA-test. ↳ indicates which model RDR targets to enhance. † is from Karpukhin et al. (2020), and ‡ is approximated from Figure 3 of Lewis et al. (2020).

Dataset	**NQ-dev**				**NQ-test**				**TriviaQA-test**			
Top-k	1	20	50	100	1	20	50	100	1	20	50	100
DPR-Single	44.2‡	76.9‡	81.3‡	84.2	46.3	78.4†	84.1	85.4†	54.4	79.4†	82.9	85.0†
↳ w/ RDR	**54.1** (+9.9)	**80.7** (+3.8)	**84.1** (+2.8)	**85.8** (+1.6)	**54.2** (+7.9)	**82.8** (+4.4)	**86.3** (+2.2)	**88.2** (+2.8)	**62.5** (+8.1)	**82.5** (+3.1)	**85.7** (+2.8)	**87.3** (+2.3)

4.2 Baseline Models and Datasets

We choose DPR (Karpukhin et al., 2020) as the main baseline to apply our proposed method to enhance retrieval performance. We run all the retrieval experiments on top of its official implementation, model weights, and evaluation scripts.[4] We also investigate the change in the end-to-end QA accuracy when RDR is used together with the readers of DPR and RAG (Lewis et al., 2020). The experiments related to RAG made use of the implementation, model weights, `FAISS` index, and evaluation scripts recently made public in huggingface/transformers (Wolf et al., 2019).[5] We consider NaturalQuestions (Kwiatkowski et al., 2019) as our main dataset because both DPR and RAG officially provides only the weights and indices for NaturalQuestions. Although we successfully reproduced and thus could report the results of experiments using DPR for TriviaQA (Joshi et al., 2017), we could not test the other settings due to the lack of training resources, e.g., size of RAM and number of available GPUs.

4.3 Training Details

Retriever To enhance DPR retriever to get RDR, we initialize the model with the pretrained DPR retriever and finetune it via knowledge distillation from DPR reader. The architecture of the retriever is thus the same with that of DPR, which consists of two bert-base encoders with the hidden dimension of 768. We set the knowledge distillation temperature to 3 and the rate of distillation loss to 1.0. We use AdamW optimizer with an initial learning rate of 1e-5 and warmup steps of 100. The other hyperparameters mostly follow the setup used to train DPR reader. We train RDR for 16 epochs with a batch size of 10 and number of passages of 16, and select the checkpoint which reports the highest retrieval accuracy on 10% or 100% of the validation set, where the former is used as an option to shorten the training time for some experiments. More details are in Appendix A.1.

Reader In order to see whether RDR can lead to the enhancement of not only retrieval recall rate but also end-to-end open-domain QA accuracy, we perform more experiments using the pretrained readers of DPR and RAG. As discussed in Section 5.2, the readers of DPR and RAG are dependent on the original retrievers they used at training time, so just replacing the retrievers with RDR during inference creates a discrepancy in the input distribution between the training and inference time. We hence finetune the reader for 3 to 6 epochs and select the model with the best Exact Match (EM) scorer on 10% or 100% of the validation set. We use the same hyperparameters to finetune DPR reader and RAG reader, except that we set the learning rate to 1e-5 for the latter and batch size to 4 to meet the budget of training resources.

5 Experimental Results

5.1 Retrieval Recall Rate

Table 1 shows the retrieval recall rate of several retrievers, measured as the percentage of the passages that contain the answer among the top-k retrieved ones, on the dev and test set of NaturalQuestions (NQ) and test set of TriviaQA. The compared models are DPR (Karpukhin et al., 2020), RAG (Lewis et al., 2020), and RDR (Ours). RDR is built on top of DPR trained with a single dataset, so we use an arrow mark to indicate that it specifically improves DPR-Single, not DPR-Multi or BM25+DPR. The results with † and ‡ are borrowed from Table 2 of the work of Karpukhin et al. (2020) and approximated from Figure 3 of the work of Lewis et al. (2020), respectively. Following Figure 3 of Lewis et al. (2020), we also show the recall of several models on the dev set of NatrualQuestions in Figure 3.

[4] https://github.com/facebookresearch/DPR
[5] https://github.com/huggingface/transformers

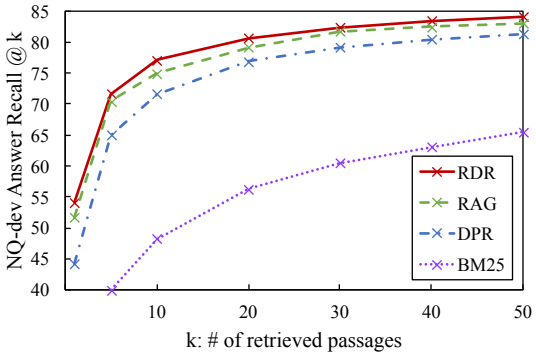


Figure 3: Answer recall rate of several models on the dev set of NQ measured as the fraction of the passages that contain the answer among the retrieved top-k ones. The values for the models other than RDR are approximated from Figure 3 of the work of Lewis et al. (2020).

Table 2: The ablation study result, which shows how the retrieval recall rate on the test set of NaturalQuestions changes when RDR enhanced from DPR-Single is trained without distillation. The recall rate shows a consistent drop at all k's when no distillation is used, and the gap is relatively large at top-1.

	DPR-Single w/ RDR	w/o distillation
Top-1	54.2	52.4 (-1.8)
Top-20	82.8	82.6 (-0.2)
Top-50	86.3	85.7 (-0.6)
Top-100	88.2	87.5 (-0.7)

Table 3: Enhancement in end-to-end QA accuracy on NaturalQuestions and TriviaQA achieved by utilizing RDR along with the readers of DPR-Single and RAG-Token. We finetune the readers for a few epochs. The values in the "Reported" column for DPR-Single-related models are the best performance achieved among $k \in \{1, 10, 20, 30, 40, 50\}$, whereas those values for RAG-Token-related models are inferred with $k = 15$, and the way to choose k follows the original inference setup of the baseline models. We could not perform RAG-related experiments on TriviaQA because the model checkpoint is not publicly available, and it was non-trivial to reproduce the with our limited computation resources.

Dataset	**NQ-test**			**TriviaQA-test**		
	Top-1	Reported		Top-1	Reported	
	EM	EM	Top-k	EM	EM	Top-k
DPR-Single	32.3	41.5	50	44.5	56.8	50
↳ w/ RDR	37.3 (+5.0)	42.1 (+0.6)	10	49.1 (+4.6)	57.0 (+0.2)	50
RAG-Token	39.4	44.1	15	-	55.2	-
↳ w/ RDR	40.9 (+1.5)	44.5 (+0.4)	15	-	-	-

As shown in the table, RDR consistently outperforms DPR-Single by a large margin throughout all datasets and the number of retrieved passages. The performance gap between RDR and DPR-Single is especially large at top-1 and when k is smaller. The significant improvement in the retrieval recall rate at small k's is especially beneficial to end-to-end open-domain QA, because it opens up more possibility to the reader to get the answer right while seeing fewer passages, which is often important for answering user's question in real-time.[6]

5.2 End-to-end Open-Domain Question Answering Accuracy

In order to see if the performance gain in passage retrieval can also lead to the enhancement in end-to-end open-domain QA accuracy, we replace the retrievers of DPR-Single and RAG-Token with RDR and measure the change in the Exact Match (EM) score. However, since the readers are trained with the outputs of their original retrievers, just replacing the retrievers with RDR at the inference time creates a gap in the input distribution between the training and inference phases. Therefore, we finetune the readers for a few epochs and report the result in Table 3.

The consistent gain in the EM score obtained with RDR suggests that the enhanced retrieval performance can also improve the end-to-end open-domain QA accuracy. The large gain in EM at reading only one passage (Top-1) would have come from the significant improvement in the top-1 retrieval recall rate. On the other hand, the improvement in the end-to-end accuracy is not proportional to the increase in retrieval performance; the gain in the former is relatively small. Our assumption is that just finetuning the reader with respect to the retriever may not be sufficient for the reader to fully benefit from the enhanced retriever, and we leave the investigation of more effective learning strategies for the reader as future work. See Appendix A.3 for other recent QA models.

[6]How the latency increases with respect to the number of passages read at inference time is reported in Appendix A.2.

Ablation Studies Table 2 reports the result when the retriever is trained without distillation. There is a consistent drop in the retrieval recall rate when distillation is not used, and the gap is relatively large at Top-1. Regarding distillation, we also tried training RDR with temperature values of $T \in \{0.5, 1, 2, 3, 4\}$ and observed from the retrieval accuracy graph on NQ-dev during training that the area under the trend line is wider with larger T.[7] Appendix A.4 shows the ablation results where the reader is not finetuned with respect to RDR, and in general, there is a drop in EM without finetuning, which may have come from the input distribution gap between the training and inference phase.

6 Conclusion

In this paper, we revisit a rather overlooked question in open-domain question answering (QA) of whether the two-tower architecture (retriever) can be entirely replaced by the one-tower architecture (reader) for accuracy if unlimited computation resources and time are allowed. Our empirical evidence indicates that the answer is no, and that the presence of the retriever seems to be essential for not only the efficiency but also the accuracy of a QA model, presumably because it becomes more robust against diverse negative documents. In the second part, given that the retriever and the reader are complementary to each other, we propose to distill the reader into the retriever to combine the best of both worlds. Our distillation method significantly enhances the recall rate of an existing retriever, which also translates into a non-trivial improvement in the end-to-end QA accuracy. Future work includes scaling up the experiment so that our method can also be evaluated on more recent models that require a large number of (e.g. 50+) GPUs to train.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*.

Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 257–264.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*.

Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. 2010. Training and testing low-degree polynomial data mappings via linear svm. *JMLR*, 11(4).

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *VLDB*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *ICML*.

John A Hartigan and Manchek A Wong. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

J. Johnson, M. Douze, and H. Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, pages 1–1.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

[7] There was no significant difference in the retrieval recall rate between the models trained with $T = 3$ and $T = 4$, so we chose $T = 3$ for all other RDR experiments.

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. In *EMNLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *TACL*.

Dik L Lee, Huei Chuang, and Kent Seamons. 1997. Document ranking and the vector-space model. *IEEE software*, 14(2):67–75.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *EMNLP*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard em approach for weakly supervised question answering. In *EMNLP*.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *EMNLP*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur P Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*.

Anshumali Shrivastava and Ping Li. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR*.

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

A Appendix

A.1 Technical Details

Batch Size and Number of Passages The original training scheme of DPR retriever uses in-batch negatives training with number of negatives of 127, which is a setup that utilizes all the passages for the other questions in the same batch as the negative passages for a question. Karpukhin et al. (2020) show that such a training scheme is effective at boosting the performance of a two-tower retriever, that DPR retriever trained with and without in-batch negatives where the number of negatives is 7 shows a large gap of 8.5 in the retrieval recall rate at $k = 5$.

On the other hand, the same setup cannot be applied for the training of reader; the reader is a one-tower model and thus needs to compute the score once for every the $(B \times B)$ pairs, unlike the two-tower retriever that can separately encode the questions and passages and use matrix multiplication to get the scores. Since the reader is the teacher to our retriever, RDR, it is trained with a smaller number of question-passage pairs at training time. As described in Section 4.3, RDR is trained with batch size of 10 and 16 passages per question, but still shows a significant improvement in retrieval performance, especially at $k = 1$.

FAISS Configuration To retrieve the top-k passages which becomes the input to the reader, we use the Maximum Inner Product Search (MIPS) index built with `FAISS` (Johnson et al., 2019). For a fair comparison, the results of the experiments related to DPR are reported using IndexFlatIP as in the work of Karpukhin et al. (2020), and those related to RAG are reported using IndexHNSWFlat with 512 neighbors to store per node, construction time search depth of 200, search depth of 128 and L2 search metric with the max norm trick (Bachrach et al., 2014). To prevent OOM while finetuning RAG reader, we additionally built and used IndexHNSWSQ index with 8bit scalar quantization. Table 4 shows the performance achieved with different types of `FAISS` index.

A.2 The Effect of Reading Fewer Passages at Inference Time

A.3 Open-Domain QA Models

In Table 6, we show the end-to-end QA accuracy of recent QA models along with that of the readers improved with RDR. The observed improvements in EM with RDR for DPR-Single and RAG-Token suggest that a higher accuracy may be achieved by applying RDR to the other recent models.

A.4 Ablation Study

Table 7 reports the results of ablation studies that the reader is not finetuned while the retriever is swapped with RDR. A drop in EM is observed in general without finetuning the reader, which may have come from the input distribution gap between the training and inference phases.

Table 4: Retrieval recall rate, index search time, and file size according to the type of `FAISS` index. The index search time is the time in seconds that takes to retrieve top-k documents for a question using the index, averaged from the search on 100 questions. Two Xeon Gold 5120 CPU cores are used to measure the search time.

	Retrieval Recall Rate				Index Search Time		
	Top-1	Top-20	Top-50	Top-100	Top-1	Top-10	File Size
IndexFlatIP	54.2	82.8	86.3	88.2	29.8	33.9	61G
IndexHNSWFlat	-0.2	-0.6	-0.6	-0.7	0.03	0.04	141G
IndexHNSWSQ	-0.2	-0.7	-0.6	-0.7	0.23	0.24	96G

Table 5: Latency of DPR reader with respect to the number of passages to read. The inference time is averaged from the runs on 100 question-passage pairs of batch size 1. Two Xeon Gold 5120 CPU cores are used across the experiments. The latency linearly increases with respect to the number of passages.

Device	Top-1	Top-10	Top-20	Top-30	Top-40	Top-50
CPU	0.63	6.76	13.49	20.17	28.9	36.1
GPU (1 × P40)	0.02	0.12	0.23	0.35	0.45	0.57

Table 6: End-to-end QA (Exact Match) accuracy of recent models. Each of the results in the left and right columns of TriviaQA is the score on the open domain test set and hidden test set, respectively.

Model	**NQ-test**	**TriviaQA-test**	
Path Retriever (Asai et al., 2020)	31.7	-	-
Graph Retriever (Min et al., 2019b)	34.7	55.8	-
Hard EM (Min et al., 2019a)	28.8	50.9	-
ORQA (Lee et al., 2019)	31.3	45.1	-
REALM (Guu et al., 2020)	38.2	-	-
DPR-Single (Karpukhin et al., 2020)	41.5	56.8	-
↳ w/ RDR	42.1 (+0.6)	57.0 (+0.2)	
BM25+DPR-Multi (Karpukhin et al., 2020)	38.8	57.9	-
SpanSeqGen (Min et al., 2020)	42.5	-	-
RAG-Token (Lewis et al., 2020)	44.1	55.2	66.1
↳ w/ RDR	44.5 (+0.4)	-	-
RAG-Sequence (Lewis et al., 2020)	44.5	56.1	68.0
T5 (Roberts et al., 2020)	36.6	-	60.5
GPT-3 few shot (Brown et al., 2020)	29.9	-	71.2
Fusion-in-Decoder (base) (Izacard and Grave, 2020)	48.2	65.0	77.1
Fusion-in-Decoder (large) (Izacard and Grave, 2020)	51.4	67.6	80.1

Table 7: Ablation result which shows the drop in the end-to-end QA accuracy (Exact Match) when no finetuning is applied to the reader while the retriever is swapped with RDR.

Dataset	**Model**	**Ablation Result**		
		Top-1	Reported	
		EM	EM	Top-k
NQ-test	DPR-Single w/ RDR	37.3	42.1	10
	w/o reader finetuning	37.2 (-0.1)	40.9 (-1.2)	10
	RAG-Token w/ RDR	40.9	44.5	15
	w/o reader finetuning	37 (-3.9)	42.9 (-1.6)	15
TriviaQA-test	DPR-Single w/ RDR	49.1	57	50
	w/o reader finetuning	49.2 (+0.1)	56.1 (-0.9)	50