Another Look at DPR: Reproduction of Training and Replication of Retrieval

Xueguang Ma(✉), Kai Sun, Ronak Pradeep, Minghan Li, and Jimmy Lin

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
{x93ma,k49sun,rpradeep,m692li,jimmylin}@uwaterloo.ca

Abstract. Text retrieval using learned dense representations has recently emerged as a promising alternative to "traditional" text retrieval using sparse bag-of-words representations. One foundational work that has garnered much attention is the dense passage retriever (DPR) proposed by Karpukhin et al. for end-to-end open-domain question answering. This work presents a reproduction and replication study of DPR. We first verify the reproducibility of the DPR model checkpoints by training passage and query encoders from scratch using two different implementations: the original code released by the authors and another independent codebase. After that, we conduct a detailed replication study of the retrieval stage, starting with model checkpoints provided by the authors but with an independent implementation from our group's Pyserini IR toolkit and PyGaggle neural text ranking library. Although our experimental results largely verify the claims of the original DPR paper, we arrive at two important additional findings: First, it appears that the original authors under-report the effectiveness of the BM25 baseline and hence also dense–sparse hybrid retrieval results. Second, by incorporating evidence from the retriever and improved answer span scoring, we manage to improve end-to-end question answering effectiveness using the same DPR models.

Keywords: Open-domain QA · Dense retrieval

1 Introduction

Reproducibility and replicability form the foundation of the scientific enterprise. Through such studies, the community gains confidence about the veracity of previously published results. These investigations are often under-valued, especially compared to work that proposes novel models, but they nevertheless make important contributions to advancing science. To be precise, throughout this paper we use the term reproducibility and replicability in the sense articulated by the ACM,[1] characterized as "different team, same experimental setup" and "different team, different experimental setup", respectively.

[1] ACM Artifact Review and Badging (Version 2.0).

M. Hagen et al. (Eds.): ECIR 2022, LNCS 13185, pp. 613–626, 2022.
https://doi.org/10.1007/978-3-030-99736-6_41

This paper focuses on a reproducibility and replicability study of the dense passage retriever (DPR) model proposed by Karpukhin et al. [8], as the authors have laid important cornerstones for end-to-end vector-based dense retrieval and open-domain question answering (QA). Specifically, we first conduct a reproduction of model training, verifying that we can obtain models with comparable levels of effectiveness using the released code from the authors as well as another implementation. Then, we conduct a replication of the retrieval pipeline, analyzing end-to-end retrieval effectiveness with our independent implementation. For a fair comparison and to reduce conflated factors, our replication study starts with the released checkpoints, as we have confirmed the reproducibility of model training during the first step.

DPR is worthy of detailed study because it represents an important exemplar of text retrieval using learned dense representations, which has emerged as a promising alternative to "traditional" text retrieval using sparse bag-of-words representations [5,11,16,19]. Our experiments largely verify the claims of Karpukhin et al. regarding the effectiveness of their proposed techniques. Moreover, we arrive at two important additional findings, one of which is inconsistent with the original work, the other of which presents an enhancement:

1. Focusing on retrieval, we find that the effectiveness of the sparse retrieval (BM25) baseline is higher than numbers reported in the original paper. Whereas they report that dense–sparse hybrid results do not meaningfully improve over dense retrieval alone, we arrive at the opposite conclusion, where hybrid techniques yield statistically significant gains. We are able to achieve on average a four-point improvement in top-20 accuracy over the best DPR results across five standard QA test collections.
2. Focusing on end-to-end QA effectiveness, we explore different techniques for evidence combination to extract the final answer span. Whereas the original DPR paper only uses scores from the reader to identify the final answer span, we investigate combining retriever scores and further experiment with the answer span selection technique described by Mao et al. [12]. In our best condition, we are able to achieve statistically significant improvements of around three points on exact match scores over the original DPR implementation while using the same exact DPR models.

To summarize, the main contribution of this work is the reproduction of DPR training and the replication of end-to-end retrieval experiments, where our experimental results add a number of important refinements to the original work. Code associated with our retrieval experiments is packaged in the Pyserini IR toolkit[2] [10] and code associated with our end-to-end QA experiments is part of the PyGaggle toolkit[3] for neural text ranking.

[2] http://pyserini.io/.
[3] http://pygaggle.ai/.

2 Methods

DPR [8] adopts a retriever–reader pipeline proposed by Chen et al. [2] for open-domain QA tasks. Both the formulation and the pipeline architecture for tackling the problem dates from at least the late 1990s [14], which means that this general approach has a long history that predates neural networks. The open-source code associated with the paper is available on GitHub (which we refer to as "the DPR repo"),[4] but it does not appear to contain code and models necessary to reproduce all results reported in the paper (more detailed discussions below).

2.1 Retriever

During retrieval, given a corpus $\mathcal{C} = \{D_1, D_2, ..., D_m\}$, the task is to return a list of the k most relevant documents (i.e., most likely to contain the answer) from $\mathcal{C}$ for each query q, where $k << |\mathcal{C}|$. In the original DPR paper and also our replication study, the corpus refers to the 2018-12-20 dump of English Wikipedia, and the "documents" are non-overlapping 100-word splits of articles.

To be clear, in most text ranking applications, the "unit of indexing" (and also retrieval) is usually referred to as a "document" D_j, although in this case it is a passage (i.e., a split) from Wikipedia. For consistency with this parlance, we use "document" and "passage" interchangeably throughout this paper. To add to the potential confusion, results of the retriever are also referred to as "contexts" that are fed to the reader.

Specifically, DPR contains a query encoder and a passage encoder, both using BERT [3] as the backbone model. Queries and passages are encoded as dense representation vectors separately as follows:

$$q^* = \text{BERT}_q(q), D_j^* = \text{BERT}_D(D_j)$$

where q^* and D_j^* are low dimensional vectors (768 dimensions by default). The relevance score of a passage to a query is computed by their vector dot product:

$$\text{Sim}(q, D_j) = \langle q^*, D_j^* \rangle$$

Thus, the retrieval problem is carried out as nearest neighbor search in dense vector space. Operationally, this is accomplished via Facebook's Faiss library [6].

During training, given a query q, a relevant passage D^+ that contains the answer, and n non-relevant passages $D_1^-, D_2^-, ...D_n^-$, the training objective is:

$$\begin{aligned}
&\mathcal{L}(q, D^+, D_1^-, D_2^-, \cdots, D_n^-) \\
&= -\log p(D = D^+ \mid Q = q) \\
&= -\log \frac{\exp(\text{Sim}(q, D_+))}{\exp(\text{Sim}(q, D^+)) + \sum_{i=1}^{n} \exp(\text{Sim}(q, D_i^-))},
\end{aligned}$$

[4] https://github.com/facebookresearch/DPR.

where $p(D = D^+ \mid Q = q)$ can be seen as a classifier given the query q evaluated at passage D^+.

Karpukhin et al. also investigated hybrid retrieval, combining results from dense retrieval (DPR) and sparse retrieval (BM25) by computing the linear combination of their respective scores to rerank the union of the two initial retrieved sets: $\lambda \cdot \text{Sim}(q, D_j) + \text{BM25}(q, D_j)$, where $\lambda = 1.1$, an empirical value tuned on the development set. BM25 retrieval was performed using Lucene with parameters $b = 0.4$ and $k_1 = 0.9$. However, the DPR repo does not appear to contain code for reproducing the BM25 and hybrid fusion results.

We attempt to replicate the retriever results reported in the DPR paper with Pyserini, an IR toolkit we have been developing since 2019 [10]. The toolkit supports sparse retrieval (i.e., BM25) via integration with another toolkit called Anserini [17] built on Lucene. Like in the original DPR work, Pyserini supports dense retrieval via integration with Facebook's Faiss library. Combining dense and sparse retrieval, the Pyserini toolkit supports hybrid retrieval as well.

Our efforts are divided into two distinct steps: First, we verify that the model checkpoints released by the DPR authors are reproducible by retraining the query and passage encoders from scratch. Then, for a fair comparison between our retrieval implementation and the original DPR work, we use the released checkpoints as the starting point of our replication study. Our retrieval implementation does not share any code with the DPR repo, other than evaluation scripts to ensure that results are comparable.

Similar to the original work, we calculate hybrid retrieval scores by linear combination of dense and sparse scores: $\text{Sim}(q, D_j) + \alpha \cdot \text{BM25}(q, D_j)$. Note that, contrary to the original work, we place the α weight on the BM25 score because this yields a more natural way to answer the pertinent research question: Given dense retrieval as a starting point, does adding BM25 as an additional relevance signal provide any value? This question is answered by comparing with a setting of $\alpha = 0$, which is equivalent to discarding BM25 results.

Finally, there are a few more details of exactly how to combine BM25 and DPR scores worth exploring. As a baseline, we use the raw scores directly in the linear combination (exactly as above). However, we notice that the range of scores from DPR and BM25 can be quite different. To potentially address this issue, we apply the following normalization technique: If a document from sparse retrieval is not in the dense retrieval results, we assign it the minimum dense retrieval score among the retrieved documents, and vice versa for the sparse retrieval score.

To arrive at a final top-k ranking, the original DPR paper generated top-k' results from DPR and top-k' results from BM25 (where $k' > k$), before considering the union of the two result sets and combining the scores to arrive at the final top-k. The original work set $k' = 2000$, but after some preliminary experimentation, we decided to fix $k' = 1000$ in our experiments since it is a more common setting in information retrieval experiments (for example, $k = 1000$ is the default in most TREC evaluations).

2.2 Reader

As is standard in a retriever–reader design, the retriever in the DPR paper returns k candidate passages (i.e., splits from Wikipedia) for each query q. The reader extracts the final answer span from the candidate contexts, where each context C_i contains the Wikipedia article title C_i^{title} and its content C_i^{text}.

The reader in DPR uses BERT-base and takes as input each candidate context C_i concatenated to the question q. Answer extraction is treated as a labeling task, and the reader identifies the answer by predicting the start and end tokens of the answer span in the contexts. To do so, the DPR reader adds a linear layer on top of BERT to predict the start logit (i.e., unnormalized probability) and end logit for each token from the final hidden layer representations. The score of an answer span is calculated by adding the start logit of the first token and the end logit of the last token. The reader returns the m highest scoring answer spans. In addition, the reader uses the learned representation of [CLS] to predict the overall relevance of the context to the question.

Mathematically, the reader operates as follows:

$$r_i, \mathcal{S} = \text{Reader}([\text{CLS}]\ q\ [\text{SEP}]\ C_i^{\text{title}} [\text{SEP}]\ C_i^{\text{text}})$$

where r_i is the overall relevance score for context C_i, and $\mathcal{S}$ comprises m potential (answer span, span score) pairs extracted from context C_i:

$$\{(S_{i,1}, s_{i,1}), (S_{i,2}, s_{i,2}), \ldots (S_{i,m}, s_{i,m})\}.$$

In the original paper, the final answer span is the candidate with the maximum span score from the context with the highest relevance score.

We attempt to replicate exactly the DPR implementation of answer extraction using our open-source PyGaggle neural reranking library, which holds the code to many of our other search-related projects. Once again, we begin with reader checkpoints released in the DPR repo, but otherwise our implementation is completely independent (other than, again, the evaluation code).

In addition to the answer extraction algorithm above, we also implement the normalized answer span scoring technique described by Mao et al. [12]. Each answer span in each candidate context C_i is re-scored according to:

$$s'_{i,j} = \text{softmax}(\vec{r})_i \cdot \text{softmax}(\vec{s_i})_j$$

where $\vec{r} = \{r_1, \cdots, r_k\}$ is the set of relevance scores of all candidate contexts and $\vec{s_i} = \{s_{i,1}, \cdots, s_{i,m}\}$ is the set of all span scores within context C_i. Duplicate answer spans across all contexts are scored by accumulating their individual scores. The answer span with the maximum score is selected as the final prediction.

In summary, we compare two answer span scoring techniques in the reader: the "original" answer span scoring technique described by Karpukhin et al. [8], and the span scoring technique described by Mao et al. [12].

2.3 Final Evidence Fusion

In the original DPR paper, the final answer span is only selected based on scores from the reader. In our replication attempt, we additionally exploit scores from the retriever to improve answer span selection. Our intuition is that predictions from both the retriever and the reader should contribute to the final answer. Concretely, instead of just using the relevance score r_i from the reader to score contexts, we fuse r_i with the retriever score R_i, calculated by: $\beta \cdot r_i + \gamma \cdot R_i$. Depending on the retrieval method, R_i can be the sparse retrieval score, the dense retrieval score, or the score after hybrid fusion. This final fused score replaces r_i as the relevance score for each context in the answer span scoring step. For example, with fusion, the answer span scoring technique from GAR [12] becomes $\text{softmax}(\beta \cdot \vec{r} + \gamma \cdot \vec{R})_i \cdot \text{softmax}(\vec{s_i})_j$.

Thus, to summarize, we explore four settings in our end-to-end QA replication: the original DPR span scoring technique, with and without retriever score fusion, and the answer span scoring technique of GAR [12], with and without retriever score fusion.

3 Experimental Setup

In this section, we clarify the models, datasets, metrics, and hyperparameters used in our experiments.

Reproduction of Training. We attempt to reproduce the DPR model checkpoints by training DPR from scratch, following the same settings in the original work as close as possible, with two different implementations. The first is the authors' released code in the DPR repo; experiments reported in the original paper used 8 × Nvidia V100 (32 GB) GPUs, as model quality depends on a large batch size (i.e., 128). The second is code from Gao et al. [4], which is based on the original implementation but exploits gradient caching to make a large batch fit on single GPU.[5] In our reproduction, we train models on 4 × V100 GPUs (the largest machine we have access to) using the authors' original code, and a single V100 GPU using the other implementation; hyperparameters are all identical to the original DPR work. The reproduced checkpoints are evaluated based on the original DPR repo's retrieval and evaluation code.

Replication of Retrieval. Our replication efforts begin with model checkpoints provided in the DPR repo. However, the authors did not release all models and datasets used in their experiments at the time of our work. Therefore, our replication experiments only use the models with released checkpoints:

- $\text{Retriever}_{\text{NQ}}$: DPR encoders trained using just the NQ dataset.
- $\text{Retriever}_{\text{Multi}}$: DPR encoders trained using a combination of datasets.

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

- $\text{Reader}_{\text{NQ-Single}}$: the DPR reader trained on NQ with negative passages from retrieval results by $\text{Retriever}_{\text{NQ}}$.
- $\text{Reader}_{\text{TQA-Multi}}$: the DPR reader trained on TriviaQA with negative passages from retrieval results by $\text{Retriever}_{\text{Multi}}$.

Datasets. We evaluate retrieval effectiveness on five standard benchmark QA datasets (NQ [9], TriviaQA [7], WQ [1], CuratedTREC [14], SQuAD [13]), exactly the same as the original paper. For end-to-end QA, we evaluate on NQ and TriviaQA with the available models. More precisely, we use the $\text{Reader}_{\text{NQ-Single}}$ model to process the retrieved contexts from $\text{Retriever}_{\text{NQ}}$ for NQ and use the $\text{Reader}_{\text{TQA-Multi}}$ model to process the retrieved contexts from $\text{Retriever}_{\text{Multi}}$ for TriviaQA.

Metrics. For retrieval, we measure effectiveness in terms of top-k retrieval accuracy, defined as the fraction of questions that have a correct answer span in the top-k retrieved contexts at least once. End-to-end QA effectiveness is measured in terms of the exact match (EM) metric, defined as the fraction of questions that have an extracted answer span exactly matching the ground truth answer. Missing from the original DPR paper, we perform significance testing to assess the statistical significance of metric differences. In all cases, we apply paired t-tests at $p < 0.01$; the Bonferroni correction is applied to correct for multiple hypothesis testing as appropriate.

Hyperparameters. In the hybrid retrieval technique described in the DPR paper, the λ weight for combining dense and sparse retrieval scores is fixed to 1.1. However, our implementation replaces λ with α (see Sect. 2.1). We tune the α values on different datasets by optimizing top-20 retrieval accuracy: For datasets where we can obtain exactly same train/dev/test splits as the original DPR paper (NQ and TriviaQA), we tune the weight on the development set. For the remaining datasets, where splits are not available or the original DPR paper does not provide specific guidance, we tune the weights on a subset of the training data. We obtain the optimal weight by performing grid search in the range $[0, 2]$ with step size 0.05.

Similarly, for final evidence fusion, we tune β (i.e., the weight for the relevance score) and γ (i.e., the weight for the retriever score) on the development set of NQ and TriviaQA using grid search. For greater computational efficiency, we perform tuning in multiple passes by interweaving a coarser step size with a finer step size. For the original DPR answer span scoring technique, we fix β to one and perform a two-step grid search on γ. We start with step size 0.05 and find the optimal γ_1. Then, we use step size 0.01 in the range $[\gamma_1 - 0.04, \gamma_1 + 0.04]$ to find the optimal γ.

For the answer span scoring technique of GAR [12], we define $\delta = \frac{\gamma}{\beta}$ and perform a three-step grid search on β and δ (i.e., the weight for the retriever score becomes $\gamma = \beta \cdot \delta$). We start with step size 0.2 for both β and δ to find the optimal pair of values β_1, δ_1. We then repeat this process with step size 0.05 and 0.01 in a smaller range around the optimal β_i and δ_i from the previous pass.

Table 1. Retrieval effectiveness comparing results from the original DPR paper ("orig") and our reproduction attempt ("repro"). The symbol * on an "orig" result indicates that the corresponding checkpoint was released.

Training	NQ		TriviaQA		WQ		Curated		SQuAD	
	top20	top100	top20	top100	top20	top100	top20	top100	top20	top100
DPR-Single (orig)	78.4*	85.4*	79.4	85.0	73.2	81.4	79.8	89.1	63.2	77.2
DPR-Single (repro)	79.1	85.9	78.9	84.5	71.0	80.2	85.1	92.2	62.1	76.8
DPR-Multi (orig)	79.4*	86.0*	78.8*	84.7*	75.0*	82.9*	89.1*	93.9*	51.6*	67.6*
DPR-Multi (repro)	79.4	87.0	78.5	84.5	75.3	83.0	88.2	94.4	58.3	72.4

For final evidence fusion, we tune the weight parameters together with the number of retrieval results (k) up to 500 with a step size of 20. Optimal parameters are selected based on the highest exact match score.

4 Results

4.1 Reproduction of Training

In Table 1, we report retrieval accuracy from our reproduced model checkpoints. DPR-Single refers to the query encoder and passage encoder trained on a single dataset only and DPR-Multi refers to the model trained on the union of NQ, TriviaQA, WQ, and CuratedTREC (with WQ and CuratedTREC up-sampled by four times given their smaller sizes). To be clear, at the time of our study, the DPR repo only released training data for NQ, TriviaQA, and SQuAD. We follow the DPR paper to prepare training data for WQ and CuratedTREC, but we prepare BM25 hard negative passages by using the Pyserini toolkit because the original repo does not contain BM25 retrieval code. The DPR-Single (repro) results are from training using the authors' original code. The DPR-Multi (repro) results are from training using the code of Gao et al. [4].

The models we train from scratch arrive at a comparable level of effectiveness to the numbers reported in the original paper. Most of the differences are relatively small, within the variability commonly seen when training neural models. Interestingly, for the DPR-Multi setting, our model appears to be quite a bit better than the original model for SQuAD.

Overall, we would consider our reproduction attempt successful. In the following experiments, to reduce the number of conflated factors, we use the DPR authors' released model checkpoints.

4.2 Replication of Retrieval

Table 2 reports top-k $= \{20, 100\}$ retrieval accuracy from our replication attempt, compared to figures copied directly from the original DPR paper; here we focus on results from $\text{Retriever}_{\text{Multi}}$. The hybrid retrieval results reported in the original DPR paper is denoted $\text{Hybrid}_{\text{orig}}$, which is not directly comparable

Table 2. Comparison between the original DPR paper ("orig") and our replication attempt ("repl"). The symbol † on a BM25 result indicates effectiveness that is significantly different from DPR. The symbol ‡ indicates that the hybrid technique is significantly better than BM25 (for SQuAD) or DPR (for all remaining collections).

Condition	top20		top100	
	orig	repl	orig	repl
NQ				
DPR	79.4	79.5	86.0	86.1
BM25	59.1	62.9†	73.7	78.3†
$\text{Hybrid}_{\text{orig}}$ ($\lambda = 1.1$)	78.0	-	83.9	-
$\text{Hybrid}_{\text{norm}}$ ($\alpha = 1.30$)	-	82.6‡	-	88.6‡
Hybrid ($\alpha = 0.55$)	-	82.7‡	-	88.1‡
TriviaQA				
DPR	78.8	78.9	84.7	84.8
BM25	66.9	76.4†	76.7	83.2†
$\text{Hybrid}_{\text{orig}}$ ($\lambda = 1.1$)	79.9	-	84.4	-
$\text{Hybrid}_{\text{norm}}$ ($\alpha = 0.95$)	-	82.6‡	-	86.5‡
Hybrid ($\alpha = 0.55$)	-	82.3‡	-	86.1‡
WQ				
DPR	75.0	75.0	82.9	83.0
BM25	55.0	62.4†	71.1	75.5†
$\text{Hybrid}_{\text{orig}}$ ($\lambda = 1.1$)	74.7	-	82.3	-
$\text{Hybrid}_{\text{norm}}$ ($\alpha = 0.95$)	-	77.1‡	-	84.4‡
Hybrid ($\alpha = 0.3$)	-	77.5‡	-	84.0‡
CuratedTREC				
DPR	89.1	88.8	93.9	93.4
BM25	70.9	80.7†	84.1	89.9†
$\text{Hybrid}_{\text{orig}}$ ($\lambda = 1.1$)	88.5	-	94.1	-
$\text{Hybrid}_{\text{norm}}$ ($\alpha = 1.05$)	-	90.1	-	95.0‡
Hybrid ($\alpha = 0.7$)	-	89.6	-	94.6‡
SQuAD				
DPR	51.6	52.0	67.6	67.7
BM25	68.8	71.1†	80.0	81.8†
$\text{Hybrid}_{\text{orig}}$ ($\lambda = 1.1$)	66.2	-	78.6	-
$\text{Hybrid}_{\text{norm}}$ ($\alpha = 2.00$)	-	75.1‡	-	84.4‡
Hybrid ($\alpha = 28$)	-	75.0‡	-	84.0‡

to either of our two techniques: $\text{Hybrid}_{\text{norm}}$ (with minimum score normalization) or Hybrid (without such normalization). We make the following observations:

First, our dense retrieval results are very close to those reported in the original paper. We consider this a successful replication attempt and our efforts add veracity to the effectiveness of the DPR technique.

Second, our Pyserini BM25 implementation outperforms the BM25 results reported in the original paper across all datasets. Furthermore, the gap is larger for $k = 20$. On average, our results represent a nearly seven-point improvement in top-20 accuracy and a nearly five-point improvement in top-100 accuracy. Since the authors of DPR have not made available their code for generating the BM25 results, we are unable to further diagnose these differences.

Nevertheless, the results do support the finding that dense retrieval using DPR is (generally) more effective than sparse retrieval. We confirm that the effectiveness differences between DPR and BM25 in our replication results are statistically significant. In all datasets except for SQuAD, DPR outperforms BM25; this is consistent with the original paper. We further confirm that for SQuAD, DPR is significantly worse than BM25. As Karpukhin et al. noted, $\text{Retriever}_{\text{Multi}}$ is trained by combining training data from all datasets but excluding SQuAD; these poor results are expected, since SQuAD draws from a very small set of Wikipedia articles.

Third, the effectiveness of hybrid dense–sparse fusion appears to be understated in the original DPR paper. Karpukhin et al. found that hybrid retrieval is *less* effective than dense retrieval in most settings, which is inconsistent with our experimental results. Instead, we find that dense–sparse retrieval consistently beats sparse retrieval across all settings. The gains from both hybrid scoring techniques are statistically significant, with the exception of top-20 for CuratedTREC. Our results might be due to better BM25 effectiveness, but we are unable to further diagnose these differences because, once again, the hybrid retrieval code is not provided in the DPR repo. Further testing also finds that the differences between the two hybrid techniques are not significant. Thus, there seems to be no strong basis to prefer one hybrid technique over the other.

Table 3. The Jaccard overlap between sparse retrieval and dense retrieval results.

Condition	$k = 20$	100	500	1000
NQ	6.1	5.2	4.4	4.2
TriviaQA	9.2	6.6	5.0	4.6
WQ	5.9	5.9	5.8	5.7
CuratedTrec	6.9	7.2	6.3	5.9
SQuAD	4.5	4.1	4.0	4.0

In Table 3, we report overlap when taking different top-k results from dense retrieval and sparse retrieval. Overlap is measured in terms of Jaccard overlap,

which is computed by the intersection over the union. It is apparent that the overlap between dense and sparse results is quite small, which suggests that they are effective in different ways. This provides an explanation of why hybrid retrieval is effective, i.e., it is exploiting different relevance signals. These results also justify the DPR design choice of retrieving $k' > k$ results from dense and sparse retrieval and then rescoring the union to arrive at the final top-k.

4.3 Replication of End-to-End QA

Table 4 presents results for our end-to-end question answering replication experiments on the NQ and TriviaQA datasets in terms of the exact match score. The original results are shown in the "orig" column. The "repl" column reports our attempt to replicate exactly the span scoring technique described in the original paper, whereas the "GAR" column shows results from using the technique proposed by Mao et al. [12]. The version of each technique that incorporates retriever scores (see Sect. 2.3) is denoted with a * symbol, i.e., "repl*" and "GAR*". For NQ, we used $\text{Retriever}_{\text{NQ}}$ and $\text{Reader}_{\text{NQ-Single}}$; for TriviaQA, we used $\text{Retriever}_{\text{Multi}}$ and $\text{Reader}_{\text{TQA-Multi}}$.

Table 4. End-to-end QA effectiveness in terms of the exact match score, comparing different answer span scoring techniques. The "orig" and "repl" columns are the original and replicated results; "GAR" refers to the technique by Mao et al. [12]; "*" represents fusion of retriever scores. The symbol † on a "repl*" result indicates sig. improvement over "repl"; on "GAR", over "repl"; on "GAR*", over "GAR". The symbol ‡ on "GAR*" indicates sig. improvement over "repl".

Condition	orig	repl	repl*	GAR	GAR*
NQ					
DPR	41.5	41.2	$42.5^{\dagger}$	41.5	$43.5^{\dagger\ddagger}$
BM25	32.6	36.3	37.0	$37.3^{\dagger}$	$38.4^{\dagger\ddagger}$
Hybrid	39.0	41.2	$43.2^{\dagger}$	$41.9^{\dagger}$	$44.0^{\dagger\ddagger}$
TriviaQA					
DPR	56.8	57.5	$58.3^{\dagger}$	$58.9^{\dagger}$	$59.5^{\dagger\ddagger}$
BM25	52.4	58.8	59.2	$61.1^{\dagger}$	$61.6^{\dagger\ddagger}$
Hybrid	57.9	59.1	$60.0^{\dagger}$	$61.0^{\dagger}$	$61.7^{\dagger\ddagger}$

With retrieval using DPR only, the "orig" and "repl" scores on both datasets are close (within a point), which suggests that we have successfully replicated the results reported in the DPR paper. With retrieval using BM25 only, our replicated results are quite a bit higher than the original DPR results; this is not a surprise given that our BM25 results are also better. When combining DPR and BM25 results at the retriever stage, the end-to-end effectiveness remains unchanged for NQ, but we observe a modest gain for TriviaQA. The

gain for TriviaQA is statistically significant. So, it is *not* the case that better top-k retrieval always leads to improvement in end-to-end effectiveness.

Comparing the "repl" and "repl*" columns, we observe that combining scores from the retriever yields modest gains across all conditions. These gains are significant for four out of the six conditions, which suggests that retriever scores contribute to improving effectiveness. Comparing the "GAR" and "repl" columns, we also observe modest gains when adopting the answer span selection technique of Mao et al. [12]. These gains are significant for all except one condition. Comparing the "GAR" and "GAR*" columns, we find that in all cases, incorporating retriever scores significantly increases effectiveness.

Finally, putting everything together—using both the answer span scoring technique of Mao et al. [12] and incorporating retriever scores—we observe statistically significant gains across all retrieval conditions, as can be seen in the "GAR*" vs. "repl" columns across all rows. Compared to the best replicated results, we obtain an improvement of approximately three points in end-to-end QA effectiveness compared to the best answer extraction approach described in the original DPR paper. Note that we are able to obtain these improvements using exactly the model checkpoints provided in the DPR repo—we have simply added two relatively simple tricks to improve scoring and evidence combination.

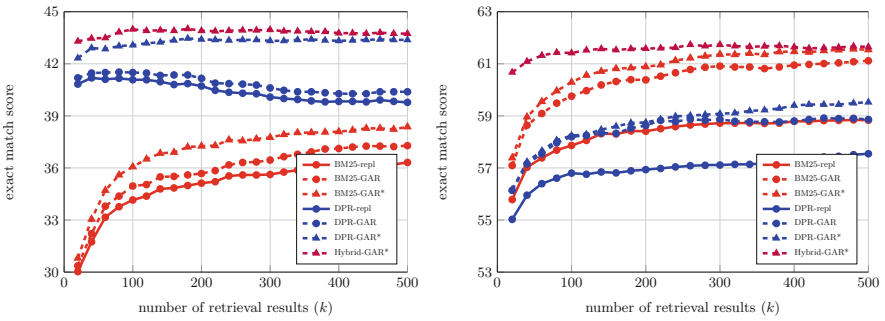

Fig. 1. End-to-end question answering effectiveness (exact match score) varying the number of retrieval results (k) for NQ (left) and TriviaQA (right).

In Fig. 1, we plot exact match scores as a function of varying k retrieval results for NQ (left) and TriviaQA (right). That is, we show how end-to-end QA effectiveness changes as the reader is provided more contexts from the retriever to consider. There are two factors here at play: On the one hand, top-k accuracy increases monotonically, i.e., as k increases, so does the likelihood that the answer appears in the contexts fed to the reader. On the other hand, the reader is asked to consider more contexts, and thus needs to discriminate the correct answer from a larger pool of candidate contexts, some of which might be low quality and thus serve as "distractors" from the correct answer. How do these factors balance out? Similar analyses in previous work with BM25 retrieval have shown

that end-to-end QA effectiveness increases with increasing k [15,18]; that is, the reader does not appear to be "confused" by the non-relevant material. Indeed, in our BM25 results we also observe the same trend.

Interestingly, however, when we switch from BM25 results to DPR results, the behavior appears to change. For TriviaQA, the effectiveness curve behaves as expected, but for NQ, the exact match score trends up and then decreases after a peak. This means that while the likelihood of the reader seeing a correct answer in the candidate contexts increases with k, it is more likely to be negatively affected by increasing amounts of non-relevant contexts as well. This general behavior is also seen for the hybrid scoring techniques: as k increases, so does the exact match score, but only up to a certain point. Beyond this point, feeding the reader more candidate contexts leads to slight decreases in end-to-end effectiveness.

5 Conclusion

The breakneck pace at which NLP and IR are advancing, we argue, makes reproducibility and replicability critical to advancing science—to ensure that we are building on a firm foundation. Our study adds to the veracity of the claims made by Karpukhin et al. [8], and our work indeed confirms that DPR is an effective dense retrieval technique. Moreover, we arrive at two important findings, one of which is inconsistent with the original work, the other of which presents an enhancement. Together, they enrich our understanding of DPR.

Acknowledgment. This research was supported in part by the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada. Computational resources were provided by Compute Ontario and Compute Canada.

References

1. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on Freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, pp. 1533–1544. Association for Computational Linguistics (2013)
2. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading Wikipedia to answer open-domain questions. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017), Vancouver, British Columbia, Canada, pp. 1870–1879 (2017)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics (2019)
4. Gao, L., Zhang, Y., Han, J., Callan, J.: Scaling deep contrastive learning batch size under memory limited setup. In: Proceedings of the 6th Workshop on Representation Learning for NLP (2021)

5. Hofstätter, S., Althammer, S., Schröder, M., Sertkan, M., Hanbury, A.: Improving efficient neural ranking models with cross-architecture knowledge distillation. arXiv:2010.02666 (2020)
6. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2021)
7. Joshi, M., Choi, E., Weld, D., Zettlemoyer, L.: 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), Vancouver, Canada, pp. 1601–1611. Association for Computational Linguistics (2017)
8. Karpukhin, V., et al.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6769–6781 (2020)
9. Kwiatkowski, T., et al.: Natural questions: a benchmark for question answering research. Trans. Assoc. Comput. Linguist. **7**, 452–466 (2019)
10. Lin, J., Ma, X., Lin, S.C., Yang, J.H., Pradeep, R., Nogueira, R.: Pyserini: a Python toolkit for reproducible information retrieval research with sparse and dense representations. In: Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021), pp. 2356–2362 (2021)
11. Lin, S.C., Yang, J.H., Lin, J.: 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), pp. 163–173 (2021)
12. Mao, Y., et al.: 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), pp. 4089–4100. Online (2021)
13. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, pp. 2383–2392 (2016)
14. Voorhees, E.M., Tice, D.M.: The TREC-8 question answering track evaluation. In: Proceedings of the Eighth Text REtrieval Conference (TREC-8), Gaithersburg, Maryland, pp. 83–106 (1999)
15. Xie, Y., et al.: Distant supervision for multi-stage fine-tuning in retrieval-based question answering. In: Proceedings of the Web Conference 2020 (WWW 2020), pp. 2934–2940 (2020)
16. Xiong, L., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: Proceedings of the 9th International Conference on Learning Representations (ICLR 2021) (2021)
17. Yang, P., Fang, H., Lin, J.: Anserini: enabling the use of Lucene for information retrieval research. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (2017)
18. Yang, W., et al.: 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), Minneapolis, Minnesota, pp. 72–77 (2019)
19. Zhan, J., Mao, J., Liu, Y., Zhang, M., Ma, S.: RepBERT: contextualized text embeddings for first-stage retrieval. arXiv:2006.15498 (2020)