PyGaggle: A Gaggle of Resources for Open-Domain Question Answering

Ronak Pradeep(✉), Haonan Chen, Lingwei Gu, Manveer Singh Tamber, and Jimmy Lin

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
{rpradeep,haonan.chen,l39gu,mtamber,jimmylin}@uwaterloo.ca

Abstract. Text retrieval using dense–sparse hybrids has been gaining popularity because of their effectiveness. Improvements to both sparse and dense models have also been noted, in the context of open-domain question answering. However, the increasing sophistication of proposed techniques places a growing strain on the reproducibility of results. Our work aims to tackle this challenge. In Generation Augmented Retrieval (GAR), a sequence-to-sequence model was used to generate candidate answer strings as well as titles of documents and actual sentences where the answer string might appear; this query expansion was applied before traditional sparse retrieval. Distilling Knowledge from Reader to Retriever (DKRR) used signals from downstream tasks to train a more effective Dense Passage Retrieval (DPR) model. In this work, we first replicate the results of GAR using a different codebase and leveraging a more powerful sequence-to-sequence model, T5. We provide tight integration with Pyserini, a popular IR toolkit, where we also add support for the DKRR-based DPR model: the combination demonstrates state-of-the-art effectiveness for retrieval in open-domain QA. To account for progress in generative readers that leverage evidence fusion for QA, so-called fusion-in-decoder (FiD), we incorporate these models into our PyGaggle toolkit. The result is a reproducible, easy-to-use, and powerful end-to-end question-answering system that forms a starting point for future work. Finally, we provide evaluation tools that better gauge whether models are generalizing or simply memorizing.

Keywords: Open-domain QA · Dense retrieval · Generative readers

1 Introduction

The past few years have seen open-domain QA progress at a breakneck pace. Given the rapid advancement of NLP and IR, both critical aspects of the overall task, this comes as little surprise. Often work needs to come along that can re-examine some of the critical papers, adding veracity to their claims and ensuring that we build on a firm foundation. For instance, Ma et al. [15] confirmed that DPR [9], a foundational work in open-domain QA, is indeed an effective

J. Kamps et al. (Eds.): ECIR 2023, LNCS 13982, pp. 148–162, 2023.
https://doi.org/10.1007/978-3-031-28241-6_10

dense retrieval approach, but further discovered that the original authors underreported the effectiveness of BM25 and thus incorrectly arrived at the conclusion that dense–sparse hybrids are worse for popular datasets like Natural Questions (NQ) [10].

As expected, significant progress has been made since the introduction of DPR in the direction of better retrievers, both sparse and dense, and better readers. Our work aims to tackle a broad set of reproducibility and replicability challenges in this context, and all our efforts result in code contributions to our group's Pyserini IR toolkit and PyGaggle open-source library.

For sparse retrieval, Generation Augmented Retrieval (GAR) trained query expansion BART models to generate answers, titles, and sentences relevant to the question. These were then used to enhance the sparse representation of the query. In this work, we train a more powerful GAR model, taking inspiration from the success of T5 [20] in similar information retrieval tasks [18]. We also examine which aspects of a GAR model contribute more to its success, from answer, title, and sentence prediction.

For dense retrieval, Izacard and Grave [4] proposed using signals from downstream reading comprehension to train a more effective DPR, called DPR_{DKRR}. We perform experiments to replicate this work, integrating this retriever into Pyserini with corresponding prebuilt indexes. Given these strong sparse and dense retrieval models, we can obtain a state-of-the-art hybrid retrieval system with little effort.

On the reader side, we integrate the popular FiD [5] model that aggregates and combines evidence from various documents to generate an answer string in a sequence-to-sequence fashion. We replicate this work in PyGaggle.

Finally, work by Lewis et al. [12] broke down popular open-domain QA benchmarks into subsets to properly examine the source of the gains, be it through training data memorization or true model generalization. To gear our systems and resources to be future-proof, we also integrate scripts that score models based on per-subset effectiveness at the retrieval and the end-to-end QA level; we are the first to consider this on the retrieval side.

To summarize, the main contribution of this work is the integration of a powerful open-domain QA system into an easy-to-use ecosystem that has tightly knit components of query expansion using our proposed, GAR-T5, dense retrieval using DPR_{DKRR}, dense–sparse hybrid retrieval, and generative reading comprehension with FiD.

Code related to our dense–sparse hybrid retrieval experiments and evaluation is packaged in Pyserini, and code related to our generative reading comprehension model as well as end-to-end QA is added to PyGaggle.

2 Data Description and Metrics

We evaluate retrieval and end-to-end QA effectiveness on the open-domain versions of Natural Questions (NQ) [10] and TriviaQA [8]. These two datasets have

Table 1. Number of instances in the different subsets of the NQ and TriviaQA test set.

Method	Total	Question overlap	Answer overlap only	No overlap
(a) NQ	3610	324	315	357
(b) TriviaQA	11313	336	411	254

emerged as the most popular datasets for evaluation, and most open-domain QA papers evaluate the effectiveness of their models on them.

However, specifically for the NQ dataset, there have been some inconsistencies in the pre-processing scripts for the queries and answers, leading to discrepancies in the retrieval effectiveness. Papers before Min et al. [17], building on DPR [9], used their topic sets, which have a few words pre-processed differently. Some that we could discern are words like "don't", "wanna", and "gonna" that took on a new life in "do n't", "wan na", and "gon na", respectively. Another was in the way quotation marks are handled. While these do not seem significant, we find that they bring about a noticeable drop in retrieval effectiveness in Sect. 4 and hence a clear spot where models have invisibly gained or dropped in effectiveness points based on the topic set used.

We integrate both these variants into Anserini and Pyserini, the DPR variants called `dpr-nq-dev` and `dpr-nq-test`, and the NQ variants called `nq-dev` and `nq-test`, the latter of which incorporates the fixes mentioned.

A detailed study by Lewis et al. [12] found that in NQ and TriviaQA, around 33% of the test queries have a near-duplicate paraphrase in their corresponding training set. Similarly, 60–70% of the answers occur somewhere in the training set answers. These observations suggest that many open-domain QA models might have memorized these as facts from training, and a thorough per-category effectiveness check is required to gain deeper insights into a model's generalization capabilities. The main subsets considered by most of the literature are:

- *Question Overlap (QO)*, where for any query, there exists some paraphrase query in the training set.
- *Answer Overlap Only (AOO)*, where there is no *question overlap*, but for the example's ground truth answer set, at least one of the answers is part of some answer set in the training set.
- *No Overlap (NO)*, where there is no train-test overlap. This set probes for open-domain QA generalization.

Table 1 reports how many instances are in each subset of the NQ and TriviaQA test set. Note that the authors only curate these annotations for around 1k examples of each test set. While Lewis et al. [12] only evaluated end-to-end QA effectiveness under these settings, we think it is critical to also look at retrieval effectiveness under this light. To this end, we integrate a tool to measure per-subset retrieval effectiveness into Pyserini and end-to-end QA effectiveness into PyGaggle. The metric used for retrieval effectiveness is the top-k accuracy, i.e.,

a score of 1 if any top-k document is relevant and 0 otherwise. We leverage the popular exact match (EM) score for end-to-end QA effectiveness.

3 Methods

The retriever–reader pipeline using neural readers was proposed by Chen et al. [1] for open-domain QA tasks. There are two modules, the retriever, which locates relevant passages given the query, and the reader, which processes the retrieved documents to generate an answer.

3.1 Retriever

Given a corpus $\mathcal{C}$ (Wikipedia segments from DPR [9]), the retriever is tasked to return a list of the k most relevant documents ($k << |\mathcal{C}|$). Here, a document is relevant if it contains one of the answer strings as a span up to normalization. In this study, the corpus refers to an English Wikipedia dump from 12/20/2018, and "documents" are non-overlapping 100-word segments of the Wikipedia articles used in DPR [9].

Karpukhin et al. [9] showed that switching the retrieval component out for *dense* representations learned by a BERT encoder (DPR) results in a significant retrieval effectiveness boost. They also investigated hybrid retrieval, combining results from dense retrieval (DPR) and sparse retrieval (BM25) but failed to find any improvements in retrieval effectiveness. However, later work [15] found that DPR under-reported the BM25 effectiveness and hence incorrectly arrived at the conclusion that hybrid retrieval *does* not result in better retrieval and end-to-end QA effectiveness. We explore how to build better sparse and dense baselines.

Sparse Methods. The focus here is Generation-Augmented Retrieval (GAR) by Mao et al. [16], a neural query expansion technique. By no means was this the first work to look into query expansion, RM3 [6] is a traditional technique employed to expand the query with terms from the top retrieved documents after an initial retrieval step. However, GAR is a powerful neural method to enhance retrieval in open-domain QA. They leveraged a pretrained sequence-to-sequence language model, BART [11], to enhance the sparse representation of the query through expansion. More specifically, they first trained three separate models that take as input the query and generate:

1. **The default target answers.** This is similar to recent work in closed-book QA where models generate answers to questions without access to external knowledge [21]. In this case, the model generates *all* possible answers separated by a special delimiter.
2. **A sentence containing the target answer.** They use as the target, the sentence from the highest BM25 ranked document (based on the original query) containing some ground-truth answer. This is an attempt to train the model to generate other words relevant to the question and target answer and help better match results with the sentence-augmented query. While this

method is not infallible to negatives, we leave further exploration to future work.

3. **Titles of passages containing any target answer.** Here, the model is trained to generate *all* titles in the top retrieved documents that contain one of the target answers. The titles are separated by a special delimiter too. Like sentences, these too might contain additional keywords or entities besides the answers that might help the augmented query pull up more relevant results.

During inference, for a particular query q, they generate three separate expanded queries q_{answer}, $q_{sentence}$, and q_{title}. Then, sparse retrieval with BM25 (default parameters) is independently performed on each query using Pyserini [14]. Finally, the top-1k results of the three queries are combined using reciprocal rank fusion (RRF) [2] to return a single ranked list. Since they provide their training code, we first try to reproduce their training and inference results. We dub this GAR-BART.

Given the success of the pretrained sequence-to-sequence transformer model T5 [20] in the similar task of document expansion for passage ranking [18], we also *replicate* their results using the powerful T5-3B model instead. Our training and generation replication uses Google's mesh-tensorflow T5 implementation.[1] We call this model GAR-T5. We train each GAR-T5 model on a Google TPU v3-8 with a constant learning rate of $1 \cdot 10^{-3}$ for 25k iterations with batches of 256. This corresponds to roughly 80 epochs on the training sets. We always use a maximum of 64 input tokens for the query and a maximum of 64, 128, and 128 output tokens for the answer, sentence, and title generation, respectively.

Dense Methods. In this subsection, we build towards the DPR_{DKRR} [4] model, a state-of-the-art dense retriever for open-domain QA retrieval. We begin with the pivotal DPR model that employs a separate query encoder and a passage encoder, both using BERT [3] as the backbone model. However, Izacard and Grave [4] only use a single encoder instead of two and signal a query by prepending the query string with "question:" and signal a document D using the template "title: D^{title} context: D^{text}". Queries and passages are encoded as dense vectors separately, and the relevance score of a query document pair is computed by their inner product. The retriever functionally performs a nearest neighbor search over the representations using Facebook's Faiss library [7].

Izacard and Grave [4] use this DPR model to initialize their retriever and leverage the Fusion-in-Decoder (FiD) reader described in Sect. 3.2. They use an iterative training pipeline where they first train a reader, use it to compute the aggregated cross-attention scores, and leverage them to finetune a retriever. Following this procedure for multiple iterations, they arrive at the more effective DPR_{DKRR} retriever.

We integrate the DPR_{DKRR} retrievers provided by the authors for both NQ and TriviaQA in Pyserini. We also provide prebuilt indexes after the corresponding retriever encodes the entire Wikipedia corpus.

[1] https://github.com/google-research/text-to-text-transfer-transformer.

Karpukhin et al. [9] and Ma et al. [15] calculate hybrid scores by linear combinations of dense and sparse scores. Here, we use reciprocal rank fusion (RRF) [2] without any task-specific tuning to fuse the GAR (sparse) and the DPR_{DKRR} (dense) ranked lists. We find that this allows us to get roughly the same effectiveness without the expensive tuning step on the development set.

3.2 Reader

As is expected in a retriever–reader paradigm, for each query q, the retriever selects k candidate documents. The reader returns the final answer span from the candidate contexts, where each document D_i contains the Wikipedia article title D_i^{title} and its content D_i^{text}.

The reader used in DPR [9] was a BERT-base and takes as input each candidate context C_i concatenated to the question q and is trained to "extract" candidate spans by predicting the start and end tokens of the target answer span. Such readers are generally called "extractive" readers.

In this paper, we use Fusion-in-Decoder (FiD) [5], a state-of-the-art generative reader that uses as backbone a generative model, T5 [20]. This model takes as input the query q and each of the top-100 retrieved documents. More precisely, each query q and document D_i are concatenated and independently run through the encoder to get an encoded representation. These representations across the ranked candidate document list are then concatenated, and the decoder *attends* over the resulting final encoding. Thus evidence fusion happens *only* in the decoder, and hence the name.

The authors experimented with both the T5-base and T5-large variants and trained them in tandem with DPR_{DKRR} as described in Sect. 3.1. They trained each of these models for 10k iterations with a batch size of 64 using 64 × V100s. This training requires significant compute beyond the scope of our paper. Inference, however, can be run on a single Tesla P40. Thus, we integrate it into PyGaggle, an inference-focused library for knowledge-intensive tasks like open-domain QA.

4 Results

4.1 Retriever

In Table 2, we report retrieval accuracy on both the NQ and TriviaQA datasets. Rows (a)–(e) showcase sparse methods, some of which we augment with query or document expansion, rows (f)–(g) denote dense ones, and rows (h)–(i) denote hybrid ones.

Row (a) represents the BM25 effectiveness using the `dpr-nq-test` topic set, which as mentioned earlier, we move away from, to the better-processed `nq-test` topic set (row (b)). This shift leads to an improvement in retrieval accuracy, highlighting the importance of using the cleaner topic set, and suggesting that previously overlooked gains may merit further examination. Hence, for the rest

Table 2. Retrieval effectiveness comparing different methods on both the NQ and TriviaQA datasets.

Method	NQ					TriviaQA				
	top5	top20	top100	top500	top1000	top5	top20	top100	top500	top1000
Sparse										
(a) BM25' [15]	43.77	62.99	78.23	85.60	88.01	–	–	–	–	–
(b) BM25	44.82	64.02	79.20	86.59	88.95	66.29	76.41	83.14	87.35	88.50
(c) BM25 + RM3	44.16	62.91	77.76	86.01	88.73	62.68	73.12	80.69	85.62	87.17
(d) GAR-BART$_{\text{RRF}}$ (repro)	61.16	76.87	86.18	91.08	92.55	71.45	79.47	85.13	88.56	89.51
(e) GAR-T5$_{\text{RRF}}$	64.62	77.17	86.90	91.63	92.91	72.82	80.66	85.95	89.07	90.06
Dense										
(f) DPR [9]	68.61	80.58	86.68	90.91	91.83	69.80	78.87	84.79	88.19	89.30
(g) DPR$_{\text{DKRR}}$	73.80	84.27	89.34	92.24	93.43	77.23	83.74	87.78	89.87	90.63
Hybrid										
(h) DPR$_{\text{Hybrid}}$ [15]	72.52	83.43	89.03	92.16	93.19	76.01	82.64	86.55	89.12	89.90
(i) RRF(GAR-T5$_{\text{RRF}}$, DPR$_{\text{DKRR}}$)	74.57	84.90	90.86	93.35	94.18	78.63	85.02	88.41	90.29	90.83

of the results, we default to using this "cleaner" topic set, deviating slightly from prior work [9,16] that used the former topic set.

Row (c) denotes BM25+RM3, a strong traditional IR baseline involving query expansion. We observe that the retrieval accuracy of BM25+RM3 (row (c)) is slightly lower than that of BM25 (row (b).

Rows (d) and (e) represent the BART and T5-based GAR models, respectively. GAR-BART$_{\text{RRF}}$ is a reproduction of the entire training and inference pipeline provided by Mao et al. [16]. We find that the retrieval effectiveness of both models is significantly higher than that of the standard BM25 method (row (b)). Comparing the two GAR models, we find that GAR-T5 performs consistently better in both datasets. This improvement is unsurprising, given that T5 is larger and pretrained on a diverse set of tasks, increasing its ability to generalize to new ones. Since our generative reader takes as input the top-100 passages, we aim to maximize the top-100 retrieval accuracy. We find that GAR-T5$_{\text{RRF}}$ achieves retrieval effectiveness on par with DPR, a dense model (row (f)).

Row (f) shows the effectiveness of the DPR retriever provided by Karpukhin et al. [9], although we use the `nq-test` topic file instead. A more detailed analysis of the effectiveness of this model is found in Ma et al. [15]. The effectiveness of DPR$_{\text{DKRR}}$ is reported in row (g). This model outperforms vanilla DPR (row (f)) across all values of retrieval effectiveness tested for both datasets. It demonstrates a solid ability to retrieve relevant documents, even in early precision settings. The strong top-5 retrieval effectiveness validates this behavior. DPR$_{\text{DKRR}}$ also outperforms all the sparse methods considered and performs consistently better than a previous hybrid model described in Ma et al. [15] (row (h)). This DPR$_{\text{Hybrid}}$ run is obtained by fusing results from DPR and BM25 using a weighted combination.

Finally, we look at our hybrid model (row (i)) that is obtained by fusing the results of GAR-T5$_{\text{RRF}}$ (row (e)) and DPR$_{\text{DKRR}}$ (row (g)). This model consistently achieves higher effectiveness than all the other methods considered across

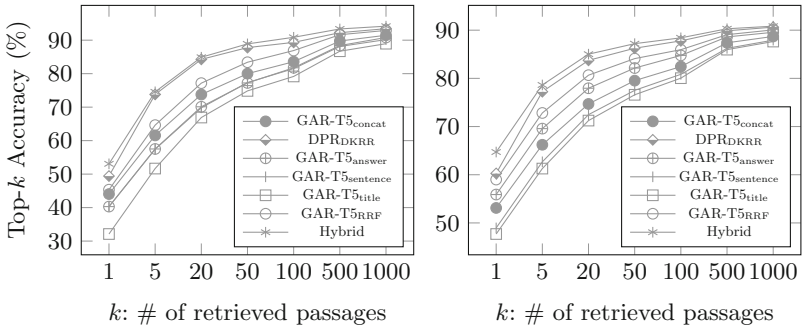

Fig. 1. Top-k accuracy of sparse, dense, and hybrid methods on NQ (left) and TriviaQA (right) as we vary the number of retrieved passages.

Table 3. Comparing the top-100 retrieval accuracy on different QA overlap subsets [12] of the test set for NQ and TriviaQA.

Condition	NQ			TriviaQA		
	QO	AOO	NO	QO	AOO	NO
(a) GAR-T5$_{\text{RRF}}$	96.30	88.57	77.59	97.92	93.19	68.50
(b) DPR$_{\text{DKRR}}$	95.06	89.84	80.95	97.32	95.13	70.87
(c) RRF(GAR-T5$_{\text{RRF}}$, DPR$_{\text{DKRR}}$)	96.91	92.38	84.03	98.21	96.10	72.44

all evaluation settings. In fact, a literature survey finds that this model has the highest retrieval accuracy for $k = 100$, the setting we want to maximize for the FiD readers. This demonstrates that hybrid methods continue to play a crucial role in knowledge-intensive tasks. Our results also suggest that sparse methods are less effective than dense methods, and we believe leveraging some learned sparse representations might bridge this gap and lead to better hybrid methods.

Figure 1 primarily shows the effectiveness of various ablations of GAR-T5. Among the individual query expansion models, GAR-T5$_{\text{answer}}$ demonstrates effectiveness on par with GAR-T5$_{\text{sentence}}$ in NQ and better in TriviaQA, both of which consistently outperform GAR-T5$_{\text{title}}$. This drop is perhaps because, in the sentences and titles cases, the model is also more prone to hallucinating additional keywords that might not be relevant to the query. Besides this, the tasks are considerably more challenging, given the lack of context provided to the model. GAR-T5$_{\text{concat}}$, which involves just a single query where the answer, sentence, and title are all appended to the original question, displays better effectiveness than each of the individual GAR-T5 models in the NQ case, but it performs worse than GAR-T5$_{\text{answer}}$ in TriviaQA. The gap between GAR-T5$_{\text{concat}}$ and GAR-T5$_{\text{RRF}}$ (where we take the RRF of GAR-T5$_{\text{answer}}$, GAR-T5$_{\text{title}}$, and GAR-T5$_{\text{sentence}}$) in the NQ test set is small, however, in the TriviaQA dataset, the gap is substantial and hence GAR-T5$_{\text{RRF}}$ stands as the most effective sparse method. The story with DPR$_{\text{DKRR}}$ and the hybrid method is the same as earlier, displaying considerably improved effectiveness compared to the sparse techniques.

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 denote the original and replicated results, respectively.

Condition	NQ		TriviaQA	
	orig	repl	orig	repl
DPR-reader [15]				
(a) DPR	43.5	–	59.5	–
(b) BM25	38.4	–	61.6	–
(c) $\text{DPR}_{\text{Hybrid}}$	44.0	–	61.7	–
FiD-base [5]				
(d) DPR_{DKRR}	50.1	49.9	69.3	68.3
(e) GAR-T5	–	49.9	–	66.8
(f) RRF(GAR-T5_{RRF}, DPR_{DKRR})	–	50.8	–	69.4
FiD-large [5]				
(g) DPR_{DKRR}	54.4	54.7	72.5	72.2
(h) GAR-T5	–	53.3	–	70.4
(i) RRF(GAR-T5_{RRF}, DPR_{DKRR})	–	55.0	–	72.9

Table 3 reports top-100 retrieval accuracy of our highest scoring sparse, dense, and hybrid models on different QA overlap subsets [12] of the NQ and TriviaQA test set. Here, we gain some more insights into our sparse and dense methods.

Row (a) presents the scores of $\text{GAR-T5}_{\text{RRF}}$, and as we can see in terms of *Question Overlap* this model performs better than our effective dense retrieval model (row (b)). As the queries in this setting are paraphrases of some of the questions in the training set, we believe that the higher scores come from the GAR-T5 models remembering the target answers, titles, and sentences and thus generating them verbatim. As a result, it would be relatively easy for a sparse lexical retrieval model which relies on exact matches to pull up relevant documents.

Row (b) presents the scores of DPR_{DKRR}, and as shown, this model demonstrates better effectiveness than $\text{GAR-T5}_{\text{RRF}}$ in terms of *Answer Overlap Only* and *No Overlap*. In these settings, since we are looking more for model generalization, it makes sense how more powerful dense methods exhibit higher effectiveness.

Finally, we look at the hybrid model in row (c), and it clearly shows the highest retrieval effectiveness across all settings. The gains are the most in the *No Overlap* subsets, the main testbed of model generalization.

4.2 Reader

Table 4 presents the end-to-end open-domain QA effectiveness in terms of the primary metric, the EM score, for both the NQ and TriviaQA test sets. Here, "orig" refers to the original results in the paper or official repository reported by the authors, and "repl" reports the results we note in our systems.

Rows (a)–(c) report results described in Ma et al. [15] using the extractive reader provided with DPR and varying the retrieval method used. Note that

Table 5. End-to-end QA effectiveness in terms of the exact match score over some of the QA Overlap subsets.

Condition	NQ			TriviaQA		
	QO	AOO	NO	QO	AOO	NO
FiD-large [5]						
(a) DPR_{DKRR}	77.2	49.5	38.1	92.0	74.5	52.4
(b) GAR-T5	76.5	45.1	36.4	92.6	70.6	52.0
(c) RRF(GAR-T5_{RRF}, DPR_{DKRR})	77.8	48.9	39.2	93.8	74.5	54.7

these were the *highest* scores obtained in the paper and involved various components that include post-processing the spans selected by the reader and tuning the weights assigned to each span. Their use of hybrid retrieval methods (row (c)) notes higher end-to-end effectiveness than using the sparse or dense models independently (rows (a), (b)).

Rows (d)–(e) denote the end-to-end effectiveness of the highest-scoring dense, sparse, and hybrid methods described paired with the FiD-base reader model and rows (f)–(h) with the FiD-large reader model. First, we note that the effectiveness of FiD integrated into PyGaggle while in a similar range does not perfectly match the original scores reported on their GitHub page.[2] This difference perhaps has to do with the different implementation choices and modifications to get the reader integrated into PyGaggle that uses HuggingFace Transformers v4.10.0 (vs. the original repository using v3.0.2).

In general, we find that leveraging dense retrieval instead of sparse retrieval results in better end-to-end open-domain QA effectiveness (rows (d) vs. (e) and rows (g) vs. (h)). This boost comes as little surprise as the top-100 retrieval effectiveness is much higher in the dense methods. However, with the FiD-base trained on NQ, this surprisingly does not bring improvements. We see similar phenomena in other knowledge-intensive tasks like multi-stage relevance ranking where improved first-stage retrieval effectiveness does not necessarily translate to better end-to-end effectiveness [19].

Similarly, hybrid retrieval shows even higher end-to-end effectiveness in all cases (rows (f) and (i)). This dominance confirms previous findings in Ma et al. [15] although, in this case, with considerably more effective sparse and dense retrievers.

Finally, moving from the FiD-base (rows (d)–(f)) to the FiD-large variant (rows (g)–(i)) results in an effectiveness boost, as one would expect with bigger language models. Our most effective system involving hybrid retrieval and the FiD-large reader (row (i)) gains 11 points compared to the comparable retriever–reader pipeline in Ma et al. [15] (row (c)) and forms an effective and easy-to-reproduce baseline for the community.

In Table 5, we report the end-to-end open-domain QA accuracy of the QA Overlap [12] subsets when using the FiD-large reader and varying the retrieval

[2] https://github.com/facebookresearch/FiD.

method. Dense generally improves over sparse retrieval (rows (a) vs. (b)), and hybrid methods display even higher effectiveness (row (c)). Most of these gains are likely from the improved first-stage retrieval of documents fed into the generative reader. The *No Overlap* subset has huge gains when moving to hybrid retrieval across the datasets. This boost could imply better model generalization capabilities. However, a surprising finding here is that the *Answer Overlap Only* subset sees a drop in effectiveness scores in the NQ case and remains the same in the TriviaQA case. A more thorough analysis of why we note such behavior is needed. Also, as expected, all methods demonstrate the highest effectiveness in the *Question Overlap* subset and the lowest effectiveness in the *No Overlap* subset.

5 Resources

In this section, we discuss the three main types of resources we provide for working with open-domain QA on Wikipedia.

5.1 GAR-T5 Query Expansions

We provide the Generation Augmented Retrieval (GAR) query expansions for NQ and TriviaQA in the widely popular HuggingFace Datasets [13]. This integration comes with various benefits - a unified interface for downloading and wrangling the expansions, single-line data loaders, and easy-to-use integration with HuggingFace Transformers [22]. These expansions for NQ and TriviaQA can be found at `castorini/nq_gar-t5_expansions` and `castorini/triviaqa_gar-t5_expansions`, respectively.

5.2 Reproduction Guides and Commands

Our reproduction guides for both the sparse and dense retrieval methods are linked on the main landing page of Pyserini.

For the main dense retriever evaluated in this paper, the DPR_{DKRR} model, we provide the retrieval and evaluation instructions for the two test sets.[3] To allow this functionality, we also convert the checkpoints provided by Izacard and Grave [4] to HuggingFace-compatible checkpoints and upload them to the HuggingFace Hub. Given these checkpoints, we encode the entire Wikipedia corpus and provide it as a prebuilt index compatible with Pyserini. Dense retrieval on this index can be performed using a single command:

```
python -m pyserini.search.faiss --topics nq-test \
    --index wikipedia-dpr-dkrr-nq \
    --encoder castorini/dkrr-dpr-nq-retriever \
    --output runs/run.nq-test.dkrr.trec --query-prefix question:
```

[3] https://github.com/castorini/pyserini/blob/master/docs/experiments-dkrr.md.

For the sparse retrieval methods considered, i.e., GAR-T5 augmented BM25, the reproduction guide for retrieval and evaluation is provided.[4] We also include instructions for hybrid retrieval using RRF [2]. Given this hybrid run, we also provide a script that can evaluate the per-subset [12] top-k retrieval effectiveness (the evaluation metric we introduced in Sect. 2). We provide a webpage with a two-click (copy and paste) reproduction matrix,[5] that provides commands for reproducing our experimental results in the current or any future version of the Pyserini toolkit. We also include a regression test script which can be easily executed using the following command from Pyserini:

```
python scripts/repro_matrix/run_all_odqa.py --topics <nq or tqa>
```

Our reproduction guides for end-to-end open-domain QA with the generative FiD reader models can be found in the PyGaggle toolkit. This toolkit is the same one where a previous study [15] integrated the extractive reader modules of DPR.

We provide instructions to run reader inference for the NQ and TriviaQA datasets.[6] We can use the following command to run reader inference given the hybrid retrieval file from Pyserini:

```
python -um pygaggle.run.evaluate_fid_reader \
    --model-name nq_reader_large \
    --retrieval-file data/run.nq-test.dkrr.gar.hybrid.json \
    --output-file data/fid_large.nq-test.dkrr.gar.hybrid.out
```

Given this FiD-large reader output file, we integrate scripts that can calculate the per-subset level end-to-end exact match (EM) scores, as described in Sect. 2.

5.3 Interactive End-to-End System

A user can query this system in an end-to-end fashion by issuing the following command:

```
python pygaggle.qa --type openbook --qa-reader fid \
    --reader-model nq_reader_base \
    --retriever-model castorini/dkrr-dpr-nq-retriever \
    --retriever-index wikipedia-dpr-dkrr-nq \
    --retriever-corpus wikipedia-dpr
```

This command pulls up an interactive demo, where a user can ask questions and have the open-domain QA model return answers. The following is an interaction with the system:

```
Enter a question: which concept album by pink floyd was adopted
into a movie
Answer:  The Wall
Enter a question: which year did the movie casablanca come out
Answer:  1942
```

[4] https://github.com/castorini/pyserini/blob/master/docs/experiments-gar-t5.md.
[5] https://castorini.github.io/pyserini/2cr/odqa.html.
[6] https://github.com/castorini/pygaggle/blob/master/docs/experiments-fid-reader.md.

6 Conclusion

We address the challenge of reproducibility in increasingly sophisticated open-domain QA systems by developing a state-of-the-art end-to-end pipeline involving a hybrid of sparse and dense retrieval and generative readers.

First, we train a more effective GAR model [16] based on T5, which expands the queries with generated predictions for answers, sentences, and titles and incorporate them into the popular open-source IR toolkit, Pyserini. Second, we replicate the results of the widely-used dense method, DPR_{DKRR} [4] retriever and provide search capabilities and prebuilt indexes through Pyserini. As a result, we can effortlessly put together a state-of-the-art hybrid retrieval system.

We incorporate the Fusion-in-Decoder generative reader into our suite of reading comprehension systems in PyGaggle. This addition allows us to provide a very effective end-to-end open-domain QA system.

The tight integration of Pyserini and PyGaggle allows users to interact directly with our systems and pose factoid questions. Following Lewis et al. [12], we also provide evaluation tools that break down effectiveness across various categories to investigate model generalization and memorization.

We believe that it will be hugely beneficial to the research community to have an effective, easy-to-use, and reproducible open-domain QA baseline and the tools to interact with these systems directly and evaluate model capabilities like the true generalization of the retriever and reader components.

Acknowledgements. This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Computational resources were provided in part by Compute Ontario and Compute Canada. We thank the Google Cloud and the TPU Research Cloud Program for credits to support some of our experimental runs.

References

1. 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)
2. Cormack, G.V., Clarke, C.L.A., Buettcher, S.: Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, pp. 758–759. Association for Computing Machinery (2009)
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), pp. 4171–4186. Minneapolis, Minnesota (2019)
4. Izacard, G., Grave, E.: Distilling knowledge from reader to retriever for question answering. ArXiv abs/2012.04584 (2021)

5. Izacard, G., Grave, E.: 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, pp. 874–880. Online, April 2021
6. Jaleel, N., et al.: UMass at TREC 2004: Novelty and HARD (2004)
7. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2021)
8. 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), pp. 1601–1611. Association for Computational Linguistics, Vancouver, Canada (2017)
9. 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)
10. Kwiatkowski, T., et al.: Natural questions: a benchmark for question answering research. Trans. Assoc. Comput. Linguist. **7**, 452–466 (2019)
11. Lewis, M., et al.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7871–7880. Online, July 2020
12. Lewis, P., Stenetorp, P., Riedel, S.: Question and answer test-train overlap in open-domain question answering datasets. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 1000–1008. Online, April 2021
13. Lhoest, Q., et al.: Datasets: a community library for natural language processing. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 175–184. Online and Punta Cana, Dominican Republic, November 2021
14. 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)
15. Ma, X., Sun, K., Pradeep, R., Li, M., Lin, J.: Another look at DPR: reproduction of training and replication of retrieval. In: Hagen, M., et al. (eds.) ECIR 2022. LNCS, vol. 13185, pp. 613–626. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99736-6_41
16. 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)
17. Min, S., et al.: NeurIPS 2020 EfficientQA competition: Systems, analyses and lessons learned. In: Escalante, H.J., Hofmann, K. (eds.) Proceedings of the NeurIPS 2020 Competition and Demonstration Track. Proceedings of Machine Learning Research, vol. 133, pp. 86–111. PMLR, 06–12 December 2021
18. Nogueira, R., Lin, J.: From doc2query to docTTTTTquery (2019)
19. Pradeep, R., Nogueira, R., Lin, J.: The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. arXiv:2101.05667 (2021)
20. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 1–67 (2020)

21. Roberts, A., Raffel, C., Shazeer, N.: 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), pp. 5418–5426. Online, November 2020
22. Wolf, T., et al.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, Online, October 2020