Pre-processing Matters! Improved Wikipedia Corpora for Open-Domain Question Answering

Manveer Singh Tamber(✉), Ronak Pradeep, and Jimmy Lin

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

Abstract. One of the contributions of the landmark Dense Passage Retriever (DPR) work is the curation of a corpus of passages generated from Wikipedia articles that have been segmented into non-overlapping passages of 100 words. This corpus has served as the standard source for question answering systems based on a retriever–reader pipeline and provides the basis for nearly all state-of-the-art results on popular open-domain question answering datasets. There are, however, multiple potential drawbacks to this corpus. First, the passages do not include tables, infoboxes, and lists. Second, the choice to split articles into non-overlapping passages results in fragmented sentences and disjoint passages that models might find hard to reason over. In this work, we experimented with multiple corpus variants from the same Wikipedia source, differing in passage size, overlapping passages, and the inclusion of linearized semi-structured data. The main contribution of our work is the replication of Dense Passage Retriever and Fusion-in-Decoder training on our corpus variants, allowing us to validate many of the findings in previous work and giving us new insights into the importance of corpus pre-processing for open-domain question answering. With better data preparation, we see improvements of over one point on both the Natural Questions dataset and the TriviaQA dataset in end-to-end effectiveness over previous work measured using the exact match score. Our results demonstrate the importance of careful corpus curation and provide the basis for future work.

Keywords: Open-domain QA · Dense retrieval · Wikipedia

1 Introduction

Dense Passage Retriever (DPR) [10] has been a pivotal work for open-domain question answering (QA). One of its contributions was the curation of a corpus of passages formed from a Wikipedia XML dump dated December 20, 2018. The authors split articles from this dump into non-overlapping passages, each 100 words long. The corpus, herein referred to as WikiText(100w), served as the textual knowledge source to allow the DPR retriever–reader pipeline to perform question answering. The corpus continues to be used in subsequent work, and improved retriever and reader models based on it have achieved state-of-the-art results on multiple open-domain QA datasets [7,8,16]. Given such community-wide adoption as the "gold standard" corpus, it is worth questioning if (and how) pre-processing choices affect the trained models.

There indeed seems to be multiple potential drawbacks to how the DPR authors pre-processed the original Wikipedia dump to form the WikiText(100w) corpus. For one, semi-structured data such as tables, infoboxes, and lists were not included in the final passage texts. The corpus contains only unstructured text from the body of the Wikipedia articles. Additionally, splitting the articles into disjoint 100-word passages results in fragmented sentences since the start or end of the passages do not necessarily align with complete sentences. Finally, the choice of disjoint passages (as opposed to overlapping passages) means that some textual information in individual passages may be isolated from associated relevant information that could be useful to provide a more complete context for question answering.

Using Wikipedia as a knowledge source in a retriever–reader pipeline has been explored earlier by Chen et al. [3]. Karpukhin et al. [10] used some of the same pre-processing code, leveraging the WikiExtractor Python library to extract cleaned text from Wikipedia articles. In our efforts, we first tried to closely replicate the corpus preparation steps from the DPR paper. For a fair comparison, we started with the same Wikipedia XML dump (from December 20, 2018) used in Karpukhin et al. [10] and we used some of the same pre-processing code. Specifically, we also used the WikiExtractor library, modifying the code to keep lists in the final cleaned text, and we used the pre-processing code from Chen et al. [3] to filter out disambiguation pages and HTML characters from the Wikipedia dump. We included additional pre-processing steps to add tables, infoboxes, and lists to our corpora and discarded unwanted leftover characters from the XML dump.

In this work, we experimented with Wikipedia corpus variants differing in passage size, overlapping passages, and the inclusion of linearized tables, infoboxes, and lists. One should note that the community has explored the inclusion of tables, infoboxes, and lists from Wikipedia in a text corpus with some success in Oğuz et al. [16], where they extracted Wikipedia tables and infoboxes exclusively from the Natural Questions dataset [11]. Here, we instead extracted the semi-structured data directly from the full Wikipedia XML dump to provide passages that are richer and contain more information. We used existing techniques to train reader and retriever models to answer questions from popular QA datasets. In particular, we started with training DPR models [10] for each corpus variant. We then trained generative Fusion-in-Decoder (FiD) reader models [8] to complete a retriever–reader pipeline.

The main contribution of this work is the replication of DPR and FiD training on our Wikipedia corpus variants. We are able to confirm many of the findings based on the original WikiText(100w) corpus, thereby adding veracity to previous results. Furthermore, we are able to increase the effectiveness of end-to-end QA using our corpus variants, highlighting the importance of corpus curation and providing a foundation for further advances.

We make our corpora and trained retriever and reader models available in HuggingFace.[1] We make our pre-processing code and retrieval code available in the Pyserini toolkit.[2] Our answer extraction code and end-to-end evaluation instructions are provided through the PyGaggle toolkit.[3]

2 Corpus Preparation

2.1 Sliding Window Segmentation

To split Wikipedia articles into passages that we feed to the retriever and reader models, we considered the use of sliding-window segmentation on a sentence level instead of splitting articles into disjoint passages of 100 words, as in the original DPR paper. Sentence-based segmentation has seen success in prior retrieval work that deals with long documents [15,17,18], and has the advantage of generating passages with natural discourse segments.

Given a Wikipedia article with any number of sentences, we employ passage size a and stride b, where $0 < b \leq a$. As a result of this segmentation, the first passage would contain the first a sentences, the second passage would start at sentence $b + 1$ and include the following a sentences, the third passage would start at sentence $2b + 1$ and contain the following a sentences, and so on. In our experiments, we considered two (a, b) configurations, $(6, 3)$ and $(8, 4)$, primarily to keep the passage lengths comparable to the WikiText(100w) corpus. We also prepended each passage with the title of the Wikipedia article to provide some global context.

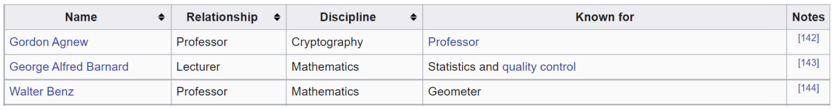
2.2 Parsing Semi-structured Data

Tables. Our goal here was to convert tables from a semi-structured format into a more linear textual form amenable to downstream readers. We also wanted the linearization to be compatible with the segmentation from Sect. 2.1 since they were likely to be interwoven with "normal" text. Hence, rows from tables were converted into a sentence-like form with the help of table headers. We began with the corresponding column header followed by a colon, then the cell's textual content followed by a comma, and then this process was repeated for each cell in the row. After each row's final cell, we ended with a period. Therefore, each row formed a "sentence". Standard segmentation choices discussed in Sect. 2.1 were applied to these linearized tables also.

It is worth noting that not every table in Wikipedia has a "standard format", where the first row holds column headers and the other rows hold cell values

1 https://huggingface.co/castorini/.
2 http://pyserini.io/.
3 http://pygaggle.ai/.

corresponding to those column headers. Some tables, which are not representative of every exception, do not have column headers and instead only have row headers. Nonetheless, our heuristic of the first row being column headers and the remaining rows having values corresponding to the column headers was adequate to linearize most tables successfully.



Name	Relationship	Discipline	Known for	Notes
Gordon Agnew	Professor	Cryptography	Professor	[142]
George Alfred Barnard	Lecturer	Mathematics	Statistics and quality control	[143]
Walter Benz	Professor	Mathematics	Geometer	[144]

```
Name: Gordon | Agnew , Relationship: Professor, Discipline: Cryptography, Known for:
Professor , Notes: , .
Name: George Alfred | Barnard , Relationship: Lecturer, Discipline: Mathematics,
Known for: Statistics and quality control , Notes: , .
Name: Walter | Benz , Relationship: Professor, Discipline: Mathematics, Known for:
Geometer, Notes: , .
```

Fig. 1. Example of a portion of a Wikipedia table (above) and our linearization into sentences (below). Note that the Wikipedia markup does not always align with the visual rendering.

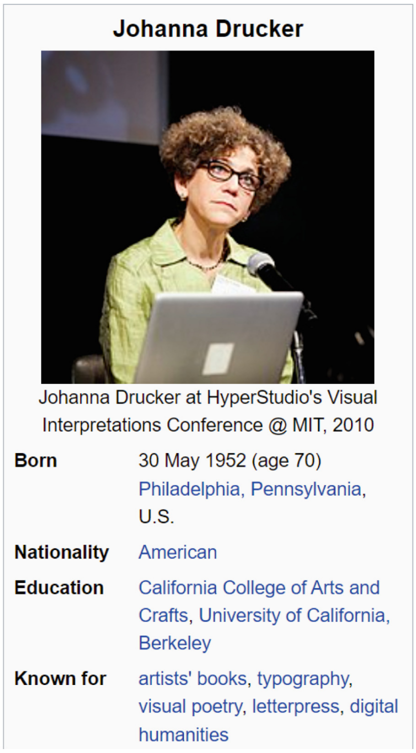
Figure 1 shows an example of how we represent tables as sentences. Given that we include the article title, the table is represented in an easy-to-consume form for both retriever and reader models to reason over, without diverging too much from the standard textual nature of "normal" passages.

Infoboxes. In Wikipedia, infoboxes are table-like content elements that usually appear at the top-right of certain articles summarizing key information about them. The XML dump presents them as labels and their corresponding data. To convert these infoboxes to passages in our corpus, much like tables, we phrased them as labels, followed by a colon, and then the corresponding data. All linearized label–data pairs were then terminated with a period. Standard segmentation choices discussed in Sect. 2.1 were also applied to infoboxes. Figure 2 shows an example; we can see that the textual form captures all the key information from the infobox.

Lists. For Wikipedia lists, we treated each list element as a separate sentence. Standard segmentation choices discussed above were also applied. Figure 3 provides an example. Given that we include the title in every passage, models can easily reason over the provided list information.

2.3 Corpus Variants and Statistics

The culmination of all these pre-processing steps and design choices was a set of Wikipedia corpus variants. While we are aware of the many combinations that can be studied, we limited ourselves to five variants that we believed were particularly worth exploring in detail, as follows:



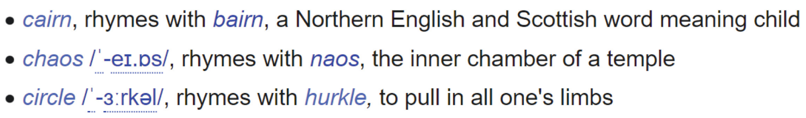
Johanna Drucker

Johanna Drucker at HyperStudio's Visual Interpretations Conference @ MIT, 2010

Born	30 May 1952 (age 70) Philadelphia, Pennsylvania, U.S.
Nationality	American
Education	California College of Arts and Crafts, University of California, Berkeley
Known for	artists' books, typography, visual poetry, letterpress, digital humanities

```
birth date: Birth date and age | 1952 | 5 | 30 .
birth place: Philadelphia, Pennsylvania , U.S..
nationality: United States | American .
field: artists’ books , typography , visual poetry , letterpress , digital humanities .
training: California College of Arts and Crafts , University of California, Berkeley.
```

Fig. 2. Example of a portion of a Wikipedia infobox (above) and our linearization into sentences (below).



- *cairn*, rhymes with *bairn*, a Northern English and Scottish word meaning child
- *chaos* /ˈ-eɪ.ɒs/, rhymes with *naos*, the inner chamber of a temple
- *circle* /ˈ-ɜːrkəl/, rhymes with *hurkle*, to pull in all one's limbs

```
"cairn", rhymes with "bairn", a Northern English and Scottish word meaning child.
"chaos" , rhymes with "naos", the inner chamber of a temple.
"circle" , rhymes with "hurkle," to pull in all one’s limbs.
```

Fig. 3. Example of a portion of a Wikipedia list (above) and our linearization into sentences (below).

Table 1. Statistics of different Wikipedia corpus variants used in this study.

Corpus	# Articles	# Passages	Avg. Len (words)
(1) WikiText(100w)	3.2 M	21.0 M	100
(2) WikiText(100w)*	5.6 M	24.0 M	95
(3) WikiText(6, 3)	5.6 M	34.0 M	109
(4) WikiAll(6, 3)	6.1 M	76.7 M	80
(5) WikiText(8, 4)	5.6 M	25.6 M	141
(6) WikiAll(8, 4)	6.1 M	57.1 M	106

- WikiText(6, 3) is a corpus without tables, infoboxes, and lists with a passage size of 6 sentences and a stride of 3 sentences.
- WikiText(8, 4) is similar to above, but with a passage size of 8 sentences and a stride of 4 sentences.
- WikiAll(6, 3) and WikiAll(8, 4) correspond to the two previous corpora, but with the inclusion of tables, infoboxes, and lists.
- The final corpus, WikiText(100w)*, was our attempt to replicate the WikiText(100w) corpus. Matching WikiText(100w), We used spaCy's `en_core_web_lg` tokenizer to count 100 words ignoring whitespace and punctuation tokens. Additionally, matching WikiText(100w), we augmented the passages at the end of articles that contained fewer than 100 words by looping back to the beginning of the article.

We performed a statistical analysis of our five different corpus variants in comparison to the original corpus; results are shown in Table 1. Comparing rows (1) and (2), we see that our replication of the WikiText(100w) corpus, referred to as WikiText(100w)*, has many more articles, a larger passage count, but the passages are (slightly) shorter than the original. This discrepancy appears to be due to articles with fewer than 100 words not being included in WikiText(100w). We argue that this choice was arbitrary as many articles were left out, and we instead elected to include these articles. Otherwise, we attempted to make WikiText(100w)* as close as possible to WikiText(100w).

Another noteworthy observation is that the corpus variants with tables, infoboxes, and lists, rows (4) and (6), have many more articles, but fewer average words per passage compared to their counterparts without the semi-structured data, rows (3) and (5). This was due to the added sentences being shorter in length, on average, compared to those from the bodies of the articles in the text-only variants.

As expected, the corpora with a passage size of 8 sentences, rows (5) and (6), have longer passages than their counterparts with a passage size of 6 sentences, rows (3) and (4). Naturally, the variants with longer passages have fewer passages overall.

3 Experimental Design

Our experiments adopted the standard retriever–reader architecture for open-domain question answering. We evaluated retrieval effectiveness on the same five datasets as Karpukhin et al. [10]: the open-domain version of Natural Questions (NQ) [11,12], the open-domain version of TriviaQA [9], WebQuestions [2], CuratedTREC [1], and SQuAD [20]. For end-to-end effectiveness, we followed Izacard and Grave [8] and only performed evaluations on the Natural Questions and the TriviaQA datasets.

3.1 Retriever Model

For each corpus variant, DPR models were trained on Google's TPU v3–8 using the Tevatron toolkit [6], with the same hyperparameters as those from Karpukhin et al. [10]. We fine-tuned the uncased variant of BERT-base with a batch size of 128 for 40 epochs, using a dropout rate of 10%, the ADAM optimizer with a learning rate of 10^{-5}, and linear scheduling with warm-up steps. We limited the question and passage lengths to 32 and 256 tokens, respectively, using the WordPiece tokenizer of BERT [5]. We considered two settings in terms of labeled data: first, only NQ; second, a multi-dataset approach that combined training data from NQ, TriviaQA, WebQuestions, and CuratedTREC.

One challenge we faced was the selection of positive and negative passages for fine-tuning the retrieval models. The original DPR paper implemented two methods: For NQ and SQuAD, each question has a corresponding span of text in Wikipedia from which the original annotators identified the answer. This span of text maps to a passage in WikiText(100w). These passages are referred to as "gold passages" and form the positive examples for each question to train the DPR model. Karpukhin et al. [10] prepared negative passages by first using BM25 for retrieval and then filtering out the positives. The authors considered these "hard negatives" because although the passages rank highly in terms of BM25 scores, they do not contain answers to the questions.

Instead of using the gold passages as the positive examples, positive passages can be selected using distant supervision. For the TREC, WebQuestions, and TriviaQA datasets, since there are no gold passages from Wikipedia, the highest-ranked passage from BM25 that contains the answer (where the search query is the concatenation of the question and the answer) is used as the single positive passage for training. Karpukhin et al. [10] prepared negative passages the same way as when gold passages are used.

To avoid matching answer spans with gold passages for each corpus, we adopted a distant supervision approach when preparing the training set for each QA dataset. In our case, the Pyserini IR toolkit [13] was used to perform BM25 retrieval using the parameters $k_1 = 0.9$ and $b = 0.4$.

After training a DPR model for each corpus, we then performed a second iteration of DPR training, inspired by Xiong et al. [21] and Oğuz et al. [16]. Again, with distant supervision, but instead of using BM25, we leveraged the already-trained DPR model to retrieve 100 passages for each query, and those that did not contain the answer were selected as the negative passages. The top-ranked passage that contained the answer was selected as the single positive passage. Using these positive and negative passages, we trained a new DPR model from scratch to obtain the second iteration DPR model.

In all these experiments, we performed dense retrieval with the Tevatron toolkit [6] and evaluated retrieval effectiveness in Pyserini [13] using the top-k accuracy metric.

3.2 Reader Model

For each corpus, we trained a FiD reader model [8] with T5-Large [19] using code made available by the authors. Hyperparameters were chosen to be consistent with Izacard and Grave [8] when computationally possible. Specifically, we used a batch size of 64, a dropout rate of 10%, and an ADAMW optimizer with a peak learning rate of 5×10^{-5}. The length of the concatenation of the question and the passage was limited to 250 tokens using the SentencePiece tokenizer of T5 [19]. Given hardware restrictions, we trained the models using 4× A100 40GB GPUs with gradient accumulation to achieve a batch size of 64. Model training was performed in mixed precision using the `bfloat16` datatype for computational efficiency. We trained the model with 10K gradient steps, which we found to be adequate, and selected the best model based on the exact match score on the validation set.

A FiD reader model was trained separately on the Natural Questions dataset and the TriviaQA dataset. On NQ, the target answer was sampled randomly from the list of answers. However, for training on TriviaQA, the unique human-generated answer was used as the target (after normalization by capitalizing the first letter of every word and leaving everything else as lowercase).

4 Results

4.1 Retrieval Results

We begin with a focus on the Natural Questions (NQ) dataset. Table 2 explores the retrieval effectiveness of DPR models trained on NQ with the different corpora discussed. Broadly, they are divided into DPR leveraging gold passages, DPR leveraging distant supervision, and DPR leveraging both distant supervision and a second round of fine-tuning.

Karpukhin et al. [10] stated that choosing positive passages using distant supervision instead of using the gold passages results in only a minor decrease in top-k accuracy for retrieval; see rows (1a) and (2f) in Table 2. We observe in row (2e) that training a DPR model on our WikiText(100w)* corpus using distantly supervised passages achieves similar top-100 effectiveness but lower top-20 effectiveness compared to WikiText(100w), row (2f). Nonetheless, using distantly supervised passages is promising and we explored other variants with this setting.

Table 2. Retrieval effectiveness comparing the use of distant supervision vs. gold passages for selecting training data, choice of segmentation, and inclusion of semi-structured data on the Natural Questions dataset.

Method	NQ-Dev		NQ-Test	
	top20	top100	top20	top100
DPR 1st round—Gold Passages				
(1a) WikiText(100w) [10]	78.1	85.0	78.4	85.4
DPR 1st round—Distantly Supervised Passages				
(2a) WikiText(6, 3)	75.9	85.0	77.0	86.5
(2b) WikiAll(6, 3)	78.1	88.2	79.3	89.1
(2c) WikiText(8, 4)	75.8	85.1	77.6	86.9
(2d) WikiAll(8, 4)	79.7	88.7	81.1	90.0
(2e) WikiText(100w)*	75.1	84.5	76.6	85.4
(2f) WikiText(100w) [10]	77.1	84.4		
DPR 2nd round—Distantly Supervised Passages				
(3a) WikiText(6, 3)	80.8	87.3	81.6	88.5
(3b) WikiAll(6, 3)	84.7	90.9	85.2	92.3
(3c) WikiText(8, 4)	81.0	87.3	82.5	89.2
(3d) WikiAll(8, 4)	85.2	91.2	86.4	92.4
(3e) WikiText(100w)*	79.6	87.1	81.2	87.8

When we trained a DPR model for the second iteration with positive and negative passages chosen using the first iteration DPR model, we observe that top-k retrieval accuracy results were higher on our WikiText(100w)* corpus than the results from both the gold and the distant supervision settings from Karpukhin et al. [10], row 3(e) vs. rows (1a) and (2f). The reason could be that better hard negatives from the first iteration DPR model (compared to BM25 hard negatives) produce a more effective final model.

Turning our attention to all the open-domain QA datasets in our study, Table 3 shows retrieval effectiveness using a DPR model trained on the amalgamation of the NQ, TriviaQA, WQ, and CuratedTREC datasets. The table begins with results across the corpus variants from BM25, standard DPR (DPR_1), and the hybrid of BM25 and DPR_1 retrieval, rows 1(a)–3(e). In the hybrid conditions, the passages are retrieved using reciprocal rank fusion (RRF) between the rankings from the DPR model and BM25. Then we have results from the two-round refined DPR (DPR_2) and the hybrid of BM25 and DPR_2 retrieval, rows 4(a)–5(e).

Except for the SQuAD dataset, the DPR models achieve higher top-k accuracy scores than BM25 retrieval, shown in rows 1(a)–(f) vs. rows 2(a)–(f) and 4(a)–(e). This observation is consistent with results from previous work [10,14]. Across the board, the second iteration DPR model seems to outperform the first iteration DPR model, shown in rows 2(a)–(e) vs. rows 4(a)–(e) of this table and rows 2(a)–(e) vs. rows 3(a)–(e) in Table 2. This finding confirms the importance of selecting negatives when fine-tuning DPR (and related) models.

Table 3. Retrieval effectiveness comparing the different corpora across the NQ, TriviaQA, WQ, CuratedTREC, and SQuAD datasets. We trained the DPR models on a combination of the NQ, TriviaQA, WQ, and CuratedTREC datasets.

Method	NQ		TriviaQA		WQ		Curated		SQuAD	
	top20	top100	top20	top100	top20	top100	top20	top100	top20	top100
BM25										
(1a) WikiText(6, 3)	64.3	78.9	77.5	84.2	62.8	76.5	81.4	91.2	74.1	84.4
(1b) WikiAll(6, 3)	66.7	81.7	78.3	84.8	64.0	78.7	80.6	91.4	72.7	83.3
(1c) WikiText(8, 4)	66.7	79.6	78.6	84.7	65.2	78.2	82.7	92.2	74.8	85.0
(1d) WikiAll(8, 4)	69.6	82.9	79.5	85.5	66.1	80.2	82.7	92.1	73.5	84.1
(1e) WikiText(100w)*	63.8	78.0	76.3	83.4	61.2	75.4	80.3	90.8	70.4	81.4
(1f) WikiText(100w) [14]	62.9	78.3	76.4	83.2	62.4	75.5	80.7	89.9	71.1	81.8
DPR_1										
(2a) WikiText(6, 3)	76.4	85.8	77.9	85.0	70.8	81.2	85.7	92.9	53.6	69.6
(2b) WikiAll(6, 3)	78.7	89.0	78.3	85.6	73.1	82.0	88.5	94.4	53.2	68.5
(2c) WikiText(8, 4)	76.9	86.3	78.5	85.4	71.8	81.3	88.0	94.7	55.1	70.7
(2d) WikiAll(8, 4)	80.4	89.8	79.0	85.7	74.1	83.1	87.9	94.0	54.4	70.2
(2e) WikiText(100w)*	75.4	85.1	78.0	84.8	70.1	80.9	88.2	94.1	50.5	67.2
(2f) WikiText(100w) [14]	79.4	87.0	78.5	84.5	75.3	83.0	88.2	94.4	58.3	72.4
$RRF(DPR_1, BM25)$										
(3a) WikiText(6, 3)	79.6	87.9	82.9	87.0	76.1	83.9	89.5	94.5	76.8	85.8
(3b) WikiAll(6, 3)	83.3	91.3	83.8	87.8	77.1	85.8	92.2	95.2	75.6	85.1
(3c) WikiText(8, 4)	81.2	88.3	83.5	87.6	75.6	84.5	91.1	95.7	77.1	86.0
(3d) WikiAll(8, 4)	84.7	91.9	84.2	88.1	78.7	85.8	91.5	95.1	76.0	85.8
(3e) WikiText(100w)*	79.6	87.8	82.4	87.0	73.8	83.5	90.1	94.7	73.3	83.1
DPR_2										
(4a) WikiText(6, 3)	81.1	88.3	81.2	86.2	76.5	84.2	90.3	94.2	60.6	75.3
(4b) WikiAll(6, 3)	85.5	91.8	81.9	87.0	80.1	87.4	91.9	95.8	60.8	75.0
(4c) WikiText(8, 4)	82.2	88.6	81.1	86.4	77.8	85.3	91.6	95.2	60.3	74.5
(4d) WikiAll(8, 4)	85.9	92.7	82.4	87.4	80.2	86.8	90.6	95.4	61.1	74.8
(4e) WikiText(100w)*	80.0	88.1	80.2	85.8	75.0	84.0	91.3	95.1	57.7	72.7
$RRF(DPR_2, BM25)$										
(5a) WikiText(6, 3)	81.6	89.2	83.3	87.4	77.4	84.7	91.6	95.4	77.9	86.5
(5b) WikiAll(6, 3)	85.3	93.0	84.2	88.0	80.3	87.5	91.5	96.4	76.9	85.9
(5c) WikiText(8, 4)	82.6	89.2	84.0	87.7	77.9	85.2	91.9	95.1	78.1	86.7
(5d) WikiAll(8, 4)	86.0	93.2	84.6	88.5	80.5	87.5	91.5	95.8	77.5	86.4
(5e) WikiText(100w)*	81.2	88.9	82.5	87.1	76.0	84.4	91.2	94.7	74.4	84.1

Also, much like in Ma et al. [14], retrieval accuracy seems to benefit from hybrid retrieval. Reciprocal rank fusion [4] between DPR ranked lists and BM25 ranked lists tend to achieve higher top-k accuracy scores than either ranking method alone, shown in rows 1(a)–(e), 2(a)–(e) vs. rows 3(a)–(e) and rows 1(a)–(e), 4(a)–(e) vs. rows 5(a)–(e).

Between the corpus variants we considered, retrieval on those with the addition of tables, infoboxes, and lists, WikiAll(6, 3) and WikiAll(8, 4), generally results in higher retrieval accuracy, shown in rows (∗a) vs. (∗b) and rows (∗c) vs. (∗d), in both Tables 2 and 3. This observation is consistent in all except for the CuratedTREC and the SQuAD datasets. Nonetheless, including these semi-structured data provides value.

Furthermore, retrieval accuracy is usually higher with WikiAll(8, 4) compared to WikiAll(6, 3), perhaps because of the longer passages, shown in rows (*d) vs.

Table 4. End-to-end QA effectiveness in terms of the exact match score. In the "DPR Ranking" condition, the passages are retrieved using the second iteration DPR model. In the "Hybrid Retrieval" condition, retrieval applied Reciprocal Rank Fusion between the rankings from the second iteration DPR model and BM25 retrieval.

Condition	NQ		TriviaQA	
	Dev	Test	Dev	Test
DPR ranking				
(1a) WikiText(6, 3)	51.0	52.4	70.6	70.7
(1b) WikiText(8, 4)	51.3	53.2	70.2	70.3
(1c) WikiAll(6, 3)	**54.3**	54.8	**71.8**	71.8
(1d) WikiAll(8, 4)	54.0	**55.0**	71.1	**72.3**
(1e) WikiText(100w)*	50.4	51.1	70.4	70.4
Hybrid ranking				
(2a) WikiText(6, 3)		53.0		72.9
(2b) WikiText(8, 4)		53.3		72.5
(2c) WikiAll(6, 3)		**55.8**		**73.7**
(2d) WikiAll(8, 4)		55.7		73.1
(2e) WikiText(100w)*		51.4		72.5

(*b). Similarly, retrieval accuracy is usually higher with WikiText(8, 4) over WikiText(6, 3), again perhaps because of the difference in average passage lengths, shown in rows (*c) vs. (*a). Retrieval accuracy tends to be the lowest on the WikiText(100w)* corpus, shown in rows (∗a, ∗b, ∗c, ∗d) vs. (∗e). These observations are generally consistent in all datasets except for CuratedTREC, where there are some exceptions.

4.2 Reader Results

Arguably, the more important measure to compare the different corpus variants is end-to-end question answering effectiveness. Retrieval evaluation is insufficient, because scores may be higher simply due to some corpora having longer passages. We evaluated end-to-end QA effectiveness in the PyGaggle toolkit using the exact match score.

Of the different corpora studied, we observe in Table 4 that exact match scores are highest in those with the inclusion of semi-structured data, seen in rows (1c), (2c), (1d), (2d). Comparing rows (∗a), (∗b), and (∗e), we see that using the text-only corpora, WikiText(6, 3) or WikiText(8, 4), results in an improvement in terms of exact match scores over the WikiText(100w)* corpus for the Natural Questions dataset, suggesting a potential superiority in segmenting articles as discussed in Sect. 2.1. Nonetheless, results for the different corpus variants are much closer on TriviaQA. It is crucial to note that answers for questions in the Natural Questions dataset were sourced from Wikipedia, whereas the same does not hold for TriviaQA, and such collection biases could be the reason for this inconsistent finding.

Table 5. Comparison to previous work for end-to-end QA effectiveness. Original results reported for FiD [8], DKRR [7], and UniK-QA [16].

Condition	NQ-test	TriviaQA-test
(1) FiD (Original) [8]	51.4	67.6
(2) DKRR [7]	54.4	72.5
(3) UniK-QA [16]	54.1	65.1
(4) WikiAll(6, 3)	**55.8**	**73.7**

Comparing the passage size and stride configurations of $(6, 3)$ vs $(8, 4)$, seen in rows (*a) vs (*b) and (*c) vs (*d), we see that results are often similar with the configurations taking turns leading.

Retrieving passages using hybrid retrieval leads to improved exact match scores, seen in rows 1(a)–(e) vs. 2(a)–(e). That is, hybrid retrieval improves not just retrieval scores but end-to-end QA effectiveness as well.

Our best experimental setting, which uses hybrid retrieval on the WikiAll(6, 3) corpus followed by a FiD-large reader model [8] to extract the answer, achieves improved effectiveness over comparable prior work also based on the FiD-large reader model. Table 5 presents these comparisons. Our best experimental setting, displayed in row (4), shows an improvement of over one point on both Natural Questions and TriviaQA.

5 Conclusion

Our paper replicates and builds on a line of work that uses a retriever–reader pipeline to process passages from the WikiText(100w) corpus to answer open-domain questions. We more thoroughly examined various techniques and strengthened findings from previous work.

We showed that not only does including tables, infoboxes, and lists in the Wikipedia corpus improve retrieval effectiveness for the Natural Questions dataset, a finding reported in Oğuz et al. [16], but that the addition clearly improves scores for also the TriviaQA and the WQ datasets. We also demonstrated that including linearized semi-structured data improves end-to-end effectiveness for both of the datasets considered: Natural Questions and TriviaQA. Splitting Wikipedia into passages in a fashion where we preserve complete sentences and allow for overlapping passages also benefits both retrieval and end-to-end effectiveness compared to the previous segmentation approach.

In terms of modeling, we find that training a DPR model for an additional iteration improves retrieval accuracy for QA, confirming Oğuz et al. [16]. Finally, following Ma et al. [14], we replicated the finding that hybrid retrieval offers better effectiveness across these new corpora. As a result of all this careful study, we have achieved the best scores that we know of for this class of models that leverages a T5-large variant of a Fusion-in-Decoder reader model to answer questions from the Natural Questions and TriviaQA datasets. Together, these improvements demonstrate the importance of careful corpus preparation and thoroughly re-examining previous work to make sure that all design variants have been explored.

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. In addition, thanks to Google Cloud and the TPU Research Cloud Program for credits to support some of our experimental runs.

References

1. Baudiš, P., Šedivý, J.: Modeling of the question answering task in the YodaQA system. In: Mothe, J., et al. (eds.) CLEF 2015. LNCS, vol. 9283, pp. 222–228. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24027-5_20
2. 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, USA, pp. 1533–1544, October 2013
3. 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 (Volume 1: Long Papers), Vancouver, Canada, pp. 1870–1879, July 2017
4. 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, SIGIR 2009, pp. 758–759. Association for Computing Machinery, New York (2009)
5. 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 (2019)
6. Gao, L., Ma, X., Lin, J., Callan, J.: Tevatron: an efficient and flexible toolkit for dense retrieval. arXiv:2203.05765 (2022)
7. Izacard, G., Grave, E.: Distilling knowledge from reader to retriever for question answering. In: International Conference on Learning Representations (2021)
8. 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
9. 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, July 2017
10. 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, November 2020. Online
11. Kwiatkowski, T., et al.: Natural questions: a benchmark for question answering research. Trans. Assoc. Comput. Linguist. **7**, 452–466 (2019)

12. Lee, K., Chang, M.W., Toutanova, K.: Latent retrieval for weakly supervised open domain question answering. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, pp. 6086–6096, July 2019
13. 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 International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2021, pp. 2356–2362. Association for Computing Machinery, New York (2021)
14. 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
15. Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 708–718. Online, November 2020
16. Oguz, B., et al.: UniK-QA: unified representations of structured and unstructured knowledge for open-domain question answering. In: Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, United States, pp. 1535–1546, July 2022
17. Pradeep, R., Li, Y., Wang, Y., Lin, J.: Neural query synthesis and domain-specific ranking templates for multi-stage clinical trial matching. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2022, pp. 2325–2330. Association for Computing Machinery, New York (2022)
18. Pradeep, R., Nogueira, R., Lin, J.J.: The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. arXiv:2101.05667 (2021)
19. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
20. 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, November 2016
21. 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)