Pruning the Index Contents for Memory Efficient Open-Domain QA

Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz

Brno University of Technology

{ifajcik,idocekal,ondrej,smrz}@fit.vutbr.cz

Abstract

This work presents a novel pipeline that demonstrates what is achievable with a combined effort of state-of-the-art approaches. Specifically, it proposes the novel R2-D2 (RANK TWICE, READ TWICE) pipeline composed of retriever, passage reranker, extractive reader, generative reader and a simple way to combine them. Furthermore, previous work often comes with a massive index of external documents that scales in the order of tens of GiB. This work presents a simple approach for pruning the contents of a massive index such that the open-domain QA system altogether with index, OS, and library components fits into 6GiB docker image while retaining only 8% of original index contents and losing only up to 3% EM accuracy[1].

1 Introduction

Recent advances in neural passage retrieval (Karpukhin et al., 2020; Izacard and Grave, 2020a; Khattab et al., 2020; Luan et al., 2021, *inter alia*) greatly improved the performance of open-domain question answering systems (open-QA). The goal of these systems is to provide an answer to factoid questions. Traditional open-QA systems (Chen et al., 2017) seek evidence for answering these questions inside the knowledge source. This is often a large corpus of short snippets of natural language, so-called passages, with information-rich contents (e.g., taken from an encyclopedia). The current state-of-the-art systems can be scaled to millions or even billions (Seo et al., 2019) of natural language passages. With the ongoing progress, and ever-growing sources of information, it can be expected that the open-QA will play a major role in everyday human life, e.g., in complementing or

even replacing document search, as we know it (Etzioni, 2011). Therefore a natural question arises: *Is all of this information relevant for current open-QA systems*?

To gain evidence towards answering this question we experiment with our simple content-based pruning approach — a binary classifier which selects whether the passage is irrelevant or not without seeing any question — on popular open-QA datasets NaturalQuestions, TriviaQA and EfficientQA. Surprisingly, we find that most (about 92%) of the information content can be pruned away with only minor (3 EM) performance degradation to be seen in the current open-domain pipelined QA systems.

As our second contribution, we present a novel pipelined open-QA baseline composed of retriever, passage reranker, extractive reader, generative reader, and a simple component fusion approach. Our system sets a new state-of-the-art on NaturalQuestions dataset. Furthermore it ended up among the top performing systems in the EfficientQA competition (Min et al., 2021)[2].

2 Pruning Approach

To reduce the size of the index, we resort to an apriori relevance classifier, which selects the relevant content without seeing a question. Note this is in contrast with the retriever, which considers a question when assigning the relevance. Consider the Wikipedia corpus split into 100-word passages. The recent work of Karpukhin et al. (2020) indicates that the distribution of golden passages — the passages containing an answer from the dataset — differs from the distribution of all passages. This is implicated by the fact that golden passages perform as better negative samples than just any randomly sampled passages when training the retriever. Therefore, given a passage p_i from

---

[1] Our demo is available at http://r2d2.fit.vutbr.cz/. Code and preprocessed data are available at https://github.com/KNOT-FIT-BUT/R2-D2.

[2] Leaderboard available at https://efficientqa.github.io/.

Wikipedia, we propose an apriori relevance classifier (we call *pruner*) into relevance class r that models the Bernoulli distribution $\boldsymbol{P}(r|p_i)$. The input of this classifier is the concatenation of Wikipedia passage (sometimes referred to as context) and its article's title separated with the special SEP token. The classifier is trained via binary cross-entropy on the set of golden passages and non-golden passages extracted from Wikipedia. In test-time, we collect the probabilities $\boldsymbol{P}(r|p_i)$ for each passage p_i in the corpus. We keep only passages p_i that satisfy the threshold constraint $\boldsymbol{P}(r|p_i) > \tau; \tau \in (0,1)$. Furthermore, we empirically verify in Section 5 that the passage embeddings from Karpukhin et al. (2020) contain strong features that capture the very same apriori relevance as this classifier does.

3 Open-QA Pipeline

To estimate the impact of corpus pruning on various open-QA components, we propose a pipelined system R2-D2 (RANK TWICE, READ TWICE). The parameters of each component are estimated separately. It is composed of DPR passage retriever (Karpukhin et al., 2020), passage reranker (see subsection 3.1), and two readers. Figure 1 shows the diagram of our system. The first reader performs an extractive span-selection similar to Fajcik et al. (2020). The second reader is based on Fusion-In-Decoder (FiD) (Izacard and Grave, 2020b).

Formally, given a question $q \in \mathcal{Q}$ from the set of all possible questions $\mathcal{Q}$ and the corpus $\mathcal{C} = \{p_1, p_2, ..., p_n\}$ composed of passages p_i, the retriever learns a ranking function $\mathrm{rank} : \mathcal{Q} \times \mathcal{C} \to \mathbb{R}$ that assigns a score to each passage. Note each passage contains its passage title.

Taking a top-K scoring passages $\mathcal{C}_r$, reranker again re-scores $\mathcal{C}_r$ scoring passages by learning a reranking function $\mathrm{rerank} : \mathcal{Q} \times \mathcal{C}_r \to \mathbb{R}$. Note that while rank and rerank have similar signatures, the computational cost of rerank over the same amount of passages is drastically higher, as it computes fine-grained interaction between tokens of question and passage.

Next, we experiment with two readers: the extractive reader reads top-V passages $\mathcal{C}_{rr}$ independently and assigns probability $\boldsymbol{P}_e(a_e|q, \mathcal{C}_{rr})$ to each span a_e in the passages (see subsection 3.2). The FiD generative reader reads top-V_2 passages $\mathcal{C}'_{rr}$ and generates an answer from probability space $\boldsymbol{P}_g(a_g|q, \mathcal{C}'_{rr})$ via greedy search.
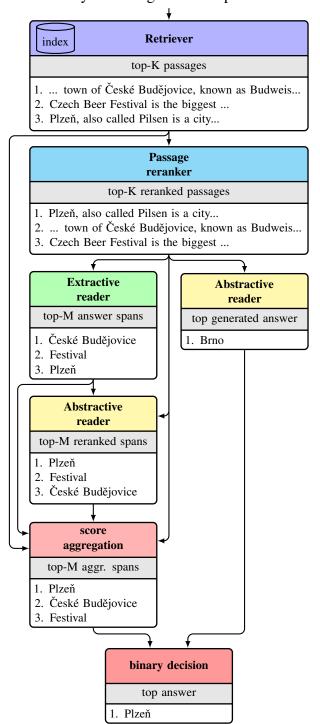
Finally, R2-D2 aggregates the outputs from all

Figure 1: R2-D2 pipeline.

components using two fusions (described in subsection 3.3).

3.1 Passage Reranker

The proposed passage reranker is based on transformer cross-encoder similar to Nogueira and Cho (2019); Luan et al. (2021). The input is the concatenation of question $q \in \mathcal{Q}$ and passage $p \in \mathcal{C}_r$ with a special SEP token between them. The passage consists of a title and context that are prepended with special start tokens and concatenated together. We denote the contextual representation of input token w obtained by the cross-encoder as $\text{En}(p, q)[w] \in \mathbb{R}^d$.

Now we can define the reranking function to the re-score passage as

$$\text{rerank}(q, p) = \text{En}(p, q)[\texttt{CLS}]^\top w \qquad (1)$$

where $w \in \mathbb{R}^d$ is a trainable vector and CLS is the special token added at the start of an input sequence. Finally, we define the following formula[3]

$$\boldsymbol{P}_{rr}(p|q, C_r) = \underset{p \in \mathcal{C}_r}{\text{softmax}} \left(\text{rerank}(q, p)\right)_p \qquad (2)$$

to assign a probability to the case that passage p contains answer to the question q.

Training. The model input for each question is exactly one positive sample supplemented with hard negatives from retriever. The ground truth passage, annotated the same way as in Karpukhin et al. (2020), is primarily used as a positive sample. If the ground truth is unknown, the positive sample is the best retriever passage containing the answer. The hard negatives are uniformly sampled from retriever top-k results that do not contain the answer. The used loss function is the cross-entropy.

3.2 Extractive Reader

Extractive reader estimates the probability $\boldsymbol{P}_e(a_e|q, \mathcal{C}_{rr})$. It is the probability of a span a_e from top-V passage $p \in \mathcal{C}_{rr}$ being an answer to a question q. We decompose the $\boldsymbol{P}_e(a_e|q, \mathcal{C}_{rr})$ into four probabilities of:

- token s being starting token of an answer span,

- token e being ending token of an answer span,

- tokens s and e being boundary tokens of an answer span (Fajcik et al., 2020),

- passage p containing an answer for the question q (inner reranker) as in Karpukhin et al. (2020).

To obtain the final probability used in test-time, we multiply them all together[4]. These probabilities are defined as:

$$\boldsymbol{P}_*(*|q, \mathcal{C}_{rr}) = \text{softmax}(s_*)_i, \qquad (3)$$

where $*$ may stand for a *start*, *end*, *joint*, and a *passage*. The i is an index of a given element, and the s_* is a vector of scores for each element among all passages in $\mathcal{C}_{rr}$. So the *softmax* normalization sum goes through all the passages. On the other hand, the s_* scores are estimated by the model with just a single passage on its input (Clark and Gardner, 2018). The scores are as follows:

$$s^i_{start} = \text{En}(p, q)[s]^\top w_{start} \qquad (4)$$

$$s^i_{end} = \text{En}(p, q)[e]^\top w_{end} \qquad (5)$$

$$s^i_{joint} = (W_j \text{En}(p, q)[s] + b_j)^\top \text{En}(p, q)[e] \qquad (6)$$

$$s^i_{passage} = \text{En}(p, q)[\texttt{CLS}]^\top w_p. \qquad (7)$$

Where $w_*, b_j \in \mathbb{R}^h$, $\text{En}(p, q)[\cdot] \in \mathbb{R}^h$, and $W_j \in \mathbb{R}^{h \times h}$ are all trainable.

We omit the spans of a title and question for answer span selection. Therefore the final answer can be selected only from the context.

The following training objective with independently marginalized components is used:

$$
\begin{aligned}
&-\log \sum_{s \in starts(C_{rr})} \boldsymbol{P}_{start}(s|q, \mathcal{C}_{rr}) \\
&-\log \sum_{e \in ends(C_{rr})} \boldsymbol{P}_{end}(e|q, \mathcal{C}_{rr}) \\
&-\log \sum_{j \in boundaries(C_{rr})} \boldsymbol{P}_{joint}(j|q, \mathcal{C}_{rr}) \\
&-\log \sum_{p \in C_{rr}} \boldsymbol{P}_{passage}(p|q, \mathcal{C}_{rr}).
\end{aligned}
\qquad (8)
$$

The sums are going through target annotations (starts, ends, etc.) obtained by the distant supervision approach.

3.3 Component Fusion

To produce the final answer, R2-D2 aggregates the log-probabilities of all system components via linear combinations tuned on validation data.

Firstly, the log-probabilities of all system components for top-M answer spans proposed by the

---

[3]Formal definition of softmax over a set is described in the Apendix E.

[4]We tried decoding from the subsets of these probabilities in Apendix F not observing significant difference.

extractive reader are aggregated. Formally, assume the $\mathcal{A}_q$ is the set of top-M answer spans from $\boldsymbol{P}_e(a|q, \mathcal{C}_{rr})$ for question q. The generative model performs the **answer reranking** evaluating the log-probability of the answer spans

$$\{\log \boldsymbol{P}_g(a|q, \mathcal{C}'_{rr}) : a \in \mathcal{A}_q\}. \qquad (9)$$

Next a logistic regression loss (11) is minimized to perform **score aggregation**. It combines the scores across the R2-D2 components to maximize the correct answer span probability over dataset $\mathcal{D}$. This dataset is composed of the top-M outputs of the extractive reader with the correct answer.

$$x(a) = [\boldsymbol{P}_e(a)\ \boldsymbol{P}_g(a)\ \boldsymbol{P}_r(p_a)\ \boldsymbol{P}_{rr}(p_a)] \quad (10)$$

$$- \sum_{(\mathcal{A}_q, gt) \in \mathcal{D}} \operatorname*{softmax}_{a \in \mathcal{A}_q} \left(w^\top \log x(a) + b \right)_{gt} \quad (11)$$

Here p_a denotes the passage containing the answer span a, $\mathcal{A}_q$ is a set of proposed answer spans, gt is the correct answer span, distribution dependencies are dropped for clarity and only the logistic regression parameters w, b are tuned in this step.

Finally, we theorized the correct answer span might not always be available in the passage set $\mathcal{C}_{rr}$, but the generative reader might be able to generate the answer from its parameters and the evidence given in passages. We introduce the binary classifier, which decides whether to select the best span answer from answer aggregation step or a free-form answer generated via FiD. Given that $s_{agg}(q) = \max_{a \in \mathcal{A}_q} w^\top x(a) + b$ is the best span score and $s_g^*(q) = \log \boldsymbol{P}_g(a_q^*|q, \mathcal{C}'_{rr})$ is the log-probability of the answer a_q^* obtained via greedy decoding for question q, a classifier is trained via binary cross-entropy $BCE(l, t)$ with log-odds ratio l and target t to do the **binary decision**

$$\sum_{(e, t) \in \mathcal{D}} BCE(w^\top[s_{agg}(e); s_g^*(e)] + b, t). \quad (12)$$

Here, the training dataset $\mathcal{D}$ contains only cases where either the extractive or the abstractive prediction is correct (but not both).

4 Experimental Setup

We implement models in PyTorch (Paszke et al., 2019) using Transformers (Wolf et al., 2020). We use 12GB GPU to train the passage reranker, 48GB GPU for the generative reader, and 16x 32GB GPUs to train the extractive reader with $V = 128$

passages at its input. The inference runs on 12GB GPU. In all experiments, we used Adam optimizer with a decoupled weight decay (Loshchilov and Hutter, 2017). Our models are evaluated by two metrics:

Exact match (EM) measures the proportion of examples, for which the system prediction matched at least one annotated ground-truth answer. We use the script from Lee et al. (2019)[5].

Accuracy@K measures the proportion of examples, for which the ground-truth answer string is present in top-K retrieved passages. We match the string exactly as Karpukhin et al. (2020)[6].

4.1 Datasets and Data Pre-processing

We evaluate our models on three datasets. Their statistics are available in Table 1. To train the reranker we filter out examples, which do not contain golden passage or exact match in top-K retrieved passages. To train the extractive reader, only examples with exact match in golden passage or top-1 retrieved passage are kept. Both filtering strategies are closely described in Appendix D.

NQ-Open (Kwiatkowski et al., 2019; Lee et al., 2019) or NaturalQuestions-Open, consists of real user queries obtained from Google search engine. The maximum length of each answer is at most 5 tokens. Each training and development sample contains 1 annotated answer, while test data contain 5-way answer annotation.

TQ-Open (Joshi et al., 2017) or TriviaQA-Open consists of question-answer pairs from 14 different trivia quiz websites. Each question contains human annotated answer and a set of answer aliases gathered from Wikipedia. We use the unfiltered version.

EfficientQA (Min et al., 2021) is a dataset collected the same way as NQ-Open through 2019, and thus may contain more questions without evidence in our corpus than NQ-Open. Furthermore, it doesn't suffer from dev/test discrepancy, as it was collected for open-domain QA directly (see Appendix B in Min et al. (2020)). We use the officially released dev set for testing.

---

[5]https://cutt.ly/rkZNIer
[6]https://cutt.ly/0luNhx4

Dataset	Train	Dev	Test
NQ-Open	79,168	8,757	3,610
- filt. reranker	71,238	-	-
- filt. ext. reader	61,755	-	-
- w/ golden passage	58,876	6,515	-
TQ-Open	78,785	8,837	11,313
- filt. reranker	69,346	-	-
- filt. ext. reader	62,332	-	-
- w/ golden passage	60,413	6,760	-
EfficientQA	-	-	1,800
NQ-Golden	176,628	4,332	8,698
TQ-Golden	181,239	4,516	9,004

Table 1: Dataset statistics. The filt. lines report how many examples are kept for training the reranker (filt. reranker) and extractive reader (filt. ext. reader). The lines w/ golden passage denote how many examples from the set contain golden passage annotation. The *Golden* sets are a datasets used to estimate the pruner.

4.2 Models and Pipeline

Pruner and Pruning. We fine-tune the base version of ELECTRA (Clark et al., 2020) with a 2-layer feed-forward network on top of it (the same way as authors do it in classification tasks) as binary classifier. To train the pruner, we create training set with 2 negative passages per positive passage from dataset's training examples with golden passage annotation. The negative passages are uniformly sampled from all non-golden Wikipedia's passages. To create development and test sets for pruner, we split the subset of dataset's development set, with examples containing golden passage annotation, using a $1 : 2$ ratio. We sample only one negative passage per positive sample for development and test sets so that datasets are balanced. We further refer to these datasets as *Golden*. The procedure is same for both datasets. The system is trained via cross-entropy in 2 epochs using batch size 12 and learning rate $3 \cdot 10^{-5}$ linearly decreasing to 0. The τ threshold is tuned so that we pool top 1.7M passages to fit the 6GiB limit. We combine these relevant passages with missing golden passages from the training data, obtaining 1,702,133 passages in total for NQ-Open and EfficientQA and 1,706,676 passages for TQ-Open.

Retriever. We use BERT-based DPR from the official checkpoint[7]. Each passage is represented via 768-dimensional embedding. We use multi-

set checkpoint for TQ-Open, as the checkpoint for TQ directly isn't officialy released. We use the same knowledge corpus containing 21,015,320 passages based on 12-20-2018 Wikipedia snapshot as Karpukhin et al. (2020). In inference time, the retriever passes $K = 200$ passages $\mathcal{C}_r$ to reranker.

Passage reranker. We use the RoBERTa-base (Liu et al., 2019) and truncate the inputs to maximum length 256. The linear scheduler with 0.1 warmup proportion is used, the number of epochs is 5 and the model is validated every 40,000 optimization steps. The initial learning rate is $1.6 \cdot 10^{-4}$, batch size equals to 8 and model reranks 24 passages per question from top-400 DPR retrieved passages. During the inference, top-K retriever passages are rescored and passed to readers.

Extractive reader. The extractive reader encoder is based on pre-trained ELECTRA-large. Its inputs are truncated if they are longer than the allowed maximum size (512 tokens). During the training phase, all spans from all $p \in \mathcal{C}_r$[8] that match[9] with at least one of the known answers are selected as target annotations. Therefore the annotations might appear in the wrong context.

The extractive reader reads top 128 passages during the training phase and when it is used without the reranker. To demonstrate the effect of reranker, the reader reads only top 24 passages if the reranker is used.

We used a linear scheduler with a warmup for the first 20,000 steps for all models. The maximum number of training steps was 200,000. The model was validated every 20,000 steps, and the best checkpoint among validations was selected. The initial learning rate was $2 \cdot 10^{-5}$ and the optimization step was done after each training example.

Generative reader. We utilize T5-large (Raffel et al., 2020) and use a concatenation of question, passages and their respective titles at the Fusion-in-Decoder's input the same way as Izacard and Grave (2020a). We truncate each passage to length 250 tokens for NQ. For TQ, as questions are significantly longer, we truncate whole inputs to the same size. Following FiD for TQ, we use only human-generated answer. In training, the golden passage always comes first, if available, and we take the rest of passages as ranked from previous step up to V_2 passages. Due to the large memory

	Method	NQ	TQ	#θ
	BM25+BERT (Mao et al., 2020)	37.7	60.1	110M
	Hard EM (Min et al., 2019a)	28.1	50.9	110M
	Path Retriever (Asai et al., 2019)	32.6	-	447M
	Graph Retriever (Min et al., 2019b)	34.5	56.0	110M
	ORQA (Lee et al., 2019)	33.3	45.0	220M
	REALM (Guu et al., 2020)	40.4	-	660M
Extractive	ProQA (Xiong et al., 2020)	34.3	-	220M
	DPR (Karpukhin et al., 2020)	41.5	56.8	220M
	DPR-subset* (Min et al., 2021)	34.8	-	220M
	RDR (Yang and Seo, 2020)	42.1	57.0	110M
	GAR+DPR (Mao et al., 2020)	43.8	-	626M
	ColBERT (large) (Khattab et al., 2020)	48.2	63.2^-	440M
	RIDER (GAR+DPR) (Mao et al., 2021)	48.3	-	626M
	BM25+SSG (Mao et al., 2020)	35.3	58.6	406M
	T51.1+SSM (Roberts et al., 2020)	35.2	61.6	11B
	RAG (Lewis et al., 2020)	44.5	56.8	516M
Generative	DPR+SSG (Min et al., 2020)	42.2	-	516M
	FiD-base (Izacard and Grave, 2020b)	48.2	65.0	333M
	FiD-large (Izacard and Grave, 2020b)	51.4	67.6	848M
	FiD-large++* (Izacard et al., 2020)	53.6	71.3	848M
	FiD-large++ (Izacard et al., 2020)	54.7	**73.3**	848M
	R1-D1 (Generative)	49.9	65.4	848M
	R1-D1 (Extractive)	50.8	65.0	445M
Ours	R2-D2 (1.7M)	52.6	68.0	1.29B
	R2-D2 (21M)	**55.0**	69.9	1.29B
	R2-D2 (21M) w/ HN-DPR	**55.9**	-	1.29B

Table 2: Comparison with the state-of-the-art in EM. #θ denotes the estimated amount of model parameters. Symbol * denotes systems with pruned or compressed index. Symbol $^-$ reports the result only for smaller system with $220M$ parameters.

Figure 2: Accuracy@K on test-data of NQ-Open.

requirements of the original approach, we use only $V_2 = 25$ passages. We use the similar hyperparameters as the original work — batch size 64, learning rate $5 \cdot 10^{-5}$ but no learning rate schedule. In test time, we decode an answer via greedy decoding.

4.3 Compressing the image size

We save models and index in half-precision without significant loss of performance. Furthermore, we use off-the-shelf `ZIP`[10] compression to reduce the size of the models and the corpus. To fit the 6GiB limit, we use 100MB CentOS8 docker image[11] and we also compress python's `site-packages` to reduce the size of PyTorch.

5 Results and Analysis

Overall results. The effectiveness of our approach is compared with the state-of-the-art in Table 2. Our system composed of just the retriever and FiD reader R1-D1 (Generative) shows inferior performance compared to FiD-large. This is most likely caused by 4 times fewer passages at its input, as in
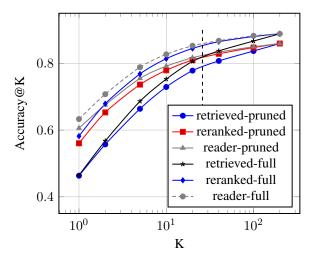
Izacard and Grave (2020b). In contrast, our ELEC-TRA based extractive reader R1-D1 (Extractive) shows large gains compared to extractive state-of-the-art, while having the same retriever as DPR. We hypothesize this may be caused by its better pre-training method, which shows strong performance through variety of tasks, but also due to training and inference with extra large input size of 128 passages and better objective. Finally, notice that our pruned system R2-D2 (1.7M) is competitive with FiD even when using just 1.7M knowledge corpus, and our full system R2-D2 (21M) is competitive even with FiD++, which uses DPR retriever improved via knowledge distillation and 26M passage corpus which also includes lists. Additionally, we evaluate our model with better retrieval model (HN-DPR) based on DPR checkpoint where hard negatives are mined using the retrieval model itself[12].

Reranker performance. Next, we analyze the performance of our retriever and reranker with Accuracy@K in Figure 2. The reranker improves the accuracy consistently for both, pruned and full version of our pipeline. Remarkably, the pruned version of our pipeline with reranker (reranked-pruned) performs better than the full version only with retriever (retrieved-full) up to $K = 26$ paragraphs. We observe the similar trend on other datasets, e.g. for TQ-Open test the reranked-pruned improves over retrieved-full up to $K = 116$ paragraphs (the analyses are in Appendix B). We also include analysis, where we rerank each passage p_i according its $s_{passage}^i$ score from extractive reader.

---

[10] https://launchpad.net/ubuntu/+source/zip
[11] `nvidia/cuda:10.2-base-centos8`

[12] https://cutt.ly/Ux5Yt4h

Passage reranking	Readers	Fusion	NQ-Open (dev)			NQ-Open			TriviaQA			EfficientQA		
			1.7M	21M	Δ	1.7M	21M	Δ	1.7M	21M	Δ	1.7M	21M	Δ
-	ext	-	46.53	48.12	-1.59	48.64	50.78	-2.14	63.18	65.01	-1.83	44.50	47.00	-2.50
-	gen	-	46.23	48.30	-2.07	48.39	49.92	-1.53	63.72	65.38	-1.66	43.50	44.83	-1.33
-	ext+gen	naive	47.05	49.14	-2.09	50.00	51.88	-1.88	64.04	66.17	-2.13	45.61	47.06	-1.45
-	ext+gen	aggr	49.05	50.87	-1.82	51.94	54.13	-2.19	65.41	67.42	-2.01	47.50	50.44	-2.94
-	ext+gen	aggr+bd	49.35	51.18	-1.83	51.88	54.07	-2.19	65.69	67.37	-1.68	47.28	49.72	-2.44
✓	ext	-	46.64	48.38	-1.74	48.92	50.72	-1.80	63.51	65.46	-1.95	45.06	47.56	-2.50
✓	gen	-	47.11	49.40	-2.29	48.31	50.69	-2.38	_67.18_	_69.14_	-1.96	45.22	47.33	-2.11
✓	ext+gen	naive	47.78	49.99	-2.21	50.33	52.44	-2.11	66.02	68.01	-1.99	46.78	49.11	-2.33
✓	ext+gen	aggr	49.89	51.80	-1.91	52.38	54.90	-2.52	66.86	68.66	-1.80	**49.44**	52.00	-2.56
✓	ext+gen	aggr+bd	**50.25**	**52.07**	-1.82	**52.58**	**54.99**	-2.41	**67.96**	**69.94**	-1.98	49.22	**52.22**	-3.00

Table 3: Ablation study. The Δ column shows the exact match difference caused by pruning.

We observe results similar to reranker for $K > 10$, indicating the reader reranks well on its own.

Pruner. Our simple pruning approach achieved 90.63% accuracy on NQ-Golden test data and 86.94% accuracy on TQ-Golden test data. This indicates that there exists a strong prior over the passages of Wikipedia in these open-domain QA datasets. Interestingly, pruner still missed 2,133/40,670 (5.2%) golden passages from the NQ-Golden training data and 6,676/50,502 (13.2%) from the TQ-Golden training data.

Memory footprint. Furthermore, we compare the memory footprint of our pruned and compressed system's docker image (pruned system) with the image of the full system on NQ-open in Figure 3. The total uncompressed size of an image is 81.01GiB while the size of the pruned image is 5.96GiB (92.6% less). Here, *codes* are python code and configurations, *corpus* is an sqlite3 database of passages, and *binaries* are the OS with python libraries. We save *dense index* as a raw h5 matrix. Interestingly, the dense corpus has a similar space requirements as the *parameters* of all 4 models used in this work.

Ablations. The ablations are listed in Table 3. We ablate results with and without using passage reranker (first column), with separate readers and their combination (second column) and with different stages of component fusion (third column). Namely, performing a *naive* answer re-ranking by generative reader means the system chooses the most probable answer span among the top-M spans provided by the extractive reader according to generative reader log-probabilities as shown in equation (9). Analogously, the *aggr* fusion denotes that the system chooses the most probable answer span according to aggregated scores, as in equation (11). Finally, the *aggr+bd* fusion denotes the binary de-
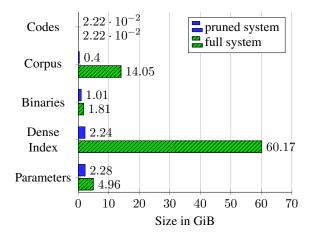

Figure 3: Component sizes inside the docker image.

cision, as shown in equation (12).

As expected, we observe that reranker improves the results consistently for generative model in all but one case (NQ-Open (1.7M)). The gains are especially large for TQ-Open (over 3 EM, underscored in Table). In fact, the results are very close or better to Izacard and Grave (2020b), suggesting that using the FiD reader with smaller context window and reranker is a reasonable alternative to memory inefficient FiD with large input size. Furthermore as expected, the extractive reader without reranker already has top-128 passages at the input, and improvements from the passage reranking are only negligible if any (less than 1 EM).

Finally, the results on NQ-Open and EfficientQA suggest applying the binary decision does not bring large improvements over the score aggregation if any. However, notice that this is not the case for TriviaQA, where the generative reader performs significantly better compared to extractive reader, suggesting both component fusions play important role in the system.

Component fusion. Furthermore, we analyze the

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	50.72	51.41	51.55	51.69
$\{g\}$	52.44	52.88	53.35	53.19
$\{e, g\}$	54.63	**55.10**	54.82	54.90

Table 4: Results for different pipeline components used for score aggregation on NQ. See text below for details.

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	52.85	53.30	53.10	52.94
$\{g\}$	52.44	52.77	53.21	53.07
$\{e, g\}$	54.35	**55.10**	54.46	54.99

Table 5: Results for binary decision on NQ for different aggregated pipeline components from Table 4.

Figure 4: Index size analysis on NQ-Open test data. Note the R2-D1 system uses retriever and reranker (R2) but only one reader (D1).

performance of each component combination in the score aggregation and its impact on the component fusion via binary decision. Both fusions are tuned on validation data and reported on test data of the NQ-Open dataset with full index. See Appendix C for analysis on additional datasets. Table 4 shows all relevant combinations of ranker *r*, reranker *rr*, extractive reader *e* and generative reader *g* probabilities used in score aggregation. In overall, we observe minor improvements up to 1EM when combining retriever and reranker scores with reader. The impact of adding a binary decision after the score aggregation is shown in Table 5. Interestingly, the binary decision component significantly improves the performance only without reranked answer scores (first row in both tables), which probably corresponds to an ensemble effect. However, fusing the generative and extractive reader via binary decision performs significantly worse on NQ-Open than fusing both readers together with score aggregation (first row in Table 5 vs. last row in Table 4). As already noted in ablations, we find this to be quite the opposite for TQ-Open. We hypothesize that the binary decision is strong in cases, where generative reader performs better to extractive reader (the case of TQ-Open).

Effect of index size. Next we analyze the effect of index size. We start by including all the golden passages from the training data (40,670 for NQ-Open, 50,502 for TQ-Open). We find the difference between using the full index and only golden passages is about 21EM (21.27 for NQ-Open, 21.01 for TQ-Open). Next, we consider adding the passages according to ranking produced by pruner. We
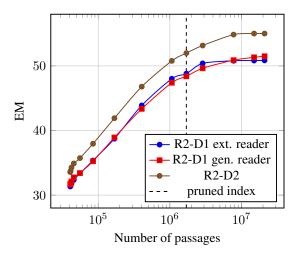
plot the system performance on NQ-Open test data as a function of index size in Figure 4. Results on TQ-Open follow the same trends and can be found in Appendix A.

Connection between retrieval embeddings and pruner. Finally, we analyze whether the DPR embeddings capture the same phenomena as our pruner does. Starting with basic statistics we compute the mean and the variance vectors of d-dimensional embeddings representing pruned (1.7M) set of documents P and those which represent the rest of the knowledge base N. We find that computing the L2 distance between mean and variance yields order of magnitude different results than the distance between randomly permuted splits of $P \cup N$ with the same size. Next we found the significant difference between average length of embedding vectors from P and N. Conclusively, we train a logistic regression classifier on balanced dataset constructed from P and subset of N, which predicts whether the passage belongs into the pruned set P or not based on its embedding. We found the classifier achieves 84.1% accuracy on the dev set, confirming our hypothesis that the apriori relevance is indeed captured in these embeddings.

6 Related Work

Pruning the document space. Min et al. (2021) presented a simple baseline which includes an index containing 1.65M passages. These include all passages from the Wikipedia articles assigned to top-5 positive passages from DPR training data for NQ (Karpukhin et al., 2020) (therefore golden pas-

sage, and highest-ranking BM25 passages to each question). However this pruning approach led to -6.7 decrease in exact match, while ours led to at most -3 EM with similar amount of passages.

Similar to our work, Izacard et al. (2020) employed three strategies to reduce the size of the index: the first is to learn a DPR encoder with embeddings projected to lower dimension, the second is to use product quantization (Gray and Neuhoff, 1998), and the third is a linear classifier (a pruner) which filters articles based on their title and list of categories. Nonetheless, their pruning approach leads to ~4 EM loss in performance with FiD-large, while still retaining 10M passages.

Dense retrieval. Lee et al. (2019) proposed the unsupervised pretraining method named inverze-cloze task. Fine-tuning such pretrained system via distant supervision surpassed the BM25 baseline for the first time in open-QA. Guu et al. (2020) demonstrated pre-training retriever and reader from scratch using an unsupervised masked language model. Xiong et al. (2020) demonstrated a pre-training method that does not require massive computational resources for unsupervised pre-training. Karpukhin et al. (2020) adopted supervised-only approach based on dual-encoder architecture, which surprisingly overtook the unsupervised approaches. Khattab et al. (2020) adopted COLBERT (Khattab and Zaharia, 2020), an approach introduced in IR that models fine-grained interaction between question and passage, for open-domain QA. Lewis et al. (2021) generated a colossal corpus of 65M questions and their respective answers. Given a question, they showed it is possible to match the state-of-the-art performance by picking an answer of the most similar question according to the learned model. Izacard and Grave (2020a) demonstrated a way of distilling FiD reader knowledge into retriever, improving its retrieval significantly, while also allowing to train retriever from scratch without any passage relevance supervision.

Passage reranker. Previous work in QA based on neural nets used bi-LSTM encoders (Wang et al., 2018; Lee et al., 2018) that score each document independently. Over time, bi-LSTM were replaced by BERT-like transformer encoders (Qiao et al., 2019; Wang et al., 2019). For document ranking, Nogueira et al. (2019) proposed a multi-stage architecture. The first stage scores each document independently, and the second estimates the more

relevant document from all document pairs. Another document ranking approach uses the seq2seq model to generate a true or false answer to the document's relevance to the query (Nogueira et al., 2020). Recent works have often focused on effective reranking. Xin et al. (2020) achieved inference speedup using early exiting, Jang and Kim (2020) proposed a smaller and faster model, and Mao et al. (2021) came up with a method which uses reader's predictions to rerank the passages. Iyer et al. (2020) marked answer predictions from reader in passages and learned to re-rank top answers along with their passage context. Our reranker is most similar to Nogueira and Cho (2019); Luan et al. (2021), except that unlike in IR, we assume there is just one correct passage and thus train our model via categorical cross-entropy.

Reader. Recent work considers two approaches towards modeling the reader — generative and extractive. The generative reader generates an answer while conditioned on question or relevant passages (Roberts et al., 2020; Lewis et al., 2020). Min et al. (2020) proposed to concatenate a question with top retriever passages as the input of pretrained seq2seq generative model. Izacard and Grave (2020b) showed its suffices to concatenate the passages in the decoder of seq2seq model, increasing the amount of top-passages the model can depend on dramatically. The extractive reader used in open-QA assumes that the answer is a continuous span string in multiple paragraphs (Chen et al., 2017). Clark and Gardner (2018) proposed to aggregate the probabilities of distantly supervised answer matches via maximum marginal likelihood (MML). Lin et al. (2018) proposed to denoise distantly supervised answer string matches in MML via paragraph-ranker. Min et al. (2019a) introduced a learning objective, which decides randomly whether to use MML objective or hard expectation-minimization via continuous annealing scheme during the training. Cheng et al. (2020) experimented with different assumptions for MML, showing improvement when marginalizing over components of span probability independently. Fajcik et al. (2020) proposed to model joint span probability directly via compound objective, instead of modeling the probability of span's start and end independently. Karpukhin et al. (2020) incorporated an independent passage classifier loss to his MML objective.

Unlike others, our work incorporates both, the generative and the extractive approach. While

our generative reader follows Izacard and Grave (2020b), our extractive reader uses a novel loss function, which includes marginalizing over target passages independently of its other components.

7 Conclusion

This work proposed R2-D2, a novel state-of-the-art pipeline for open-domain QA based on 4 components: retriever, reranker, generative reader and extractive reader. Furthermore, it proposed an approach for reducing the pipeline size to fit 6GiB Docker Image. The core idea of our approach was to drastically reduce the colossal number of passages commonly used within the knowledge-base of retrieval-based open-domain QA systems (by 92%) with only minor loss of performance (-3 EM). We believe our pipeline composed of multiple heterogeneous components is an ideal benchmark system for future research. Additionally, the pruned index size opens up new possibilities, as it now fits to most modern GPUs. However, with such a drastic reduction of knowledge-base, more questions arise: *What is it that makes the passage being apriori relevant? Does this strong prior over passages suggest that these open-domain answering datasets aren't really "open"?*. We would like to address these questions in our future research.

Acknowledgments

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2019. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

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

Hao Cheng, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Probabilistic assumptions matter: Improved models for distantly-supervised document-level question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5657–5667.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Oren Etzioni. 2011. Search needs a shake-up. *Nature*, 476(7358):25–26.

Martin Fajcik, Josef Jon, Santosh Kesiraju, and Pavel Smrz. 2020. Rethinking the objectives of extractive question answering. *arXiv preprint arXiv:2008.12804*.

Robert M. Gray and David L. Neuhoff. 1998. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Srinivasan Iyer, Sewon Min, Yashar Mehdad, and Wen-tau Yih. 2020. Reconsider: Re-ranking using span-focused cross-attention for open domain question answering. *arXiv preprint arXiv:2010.10757*.

Gautier Izacard and Edouard Grave. 2020a. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*.

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

Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Sebastian Riedel, and Edouard Grave. 2020. A memory efficient baseline for open domain question answering. *arXiv preprint arXiv:2012.15156*.

Youngjin Jang and Harksoo Kim. 2020. Document re-ranking model for machine-reading and comprehension. *Applied Sciences*, 10(21).

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2020. Relevance-guided supervision for openQA with colBERT. *arXiv preprint arXiv:2007.00814*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 565–569, Brussels, Belgium. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

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

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020. Generation-augmented retrieval for open-domain question answering. *arXiv preprint arXiv:2009.08553*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Reader-guided passage reranking for open-domain question answering. *arXiv preprint arXiv:2101.00294*.

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, et al. 2021. NeurIPS 2020 EfficientQA competition: Systems, analyses and lessons learned. *arXiv preprint arXiv:2101.00133*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2844–2857.

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

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797.

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

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. *arXiv preprint arXiv:1904.07531*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

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

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*, pages 5981–5988.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5878–5882, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early exiting BERT for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88, Online. Association for Computational Linguistics.

Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. Progressively pretrained dense corpus index for open-domain question answering. *arXiv preprint arXiv:2005.00038*.

Sohee Yang and Minjoon Seo. 2020. Is retriever merely an approximator of reader? *arXiv preprint arXiv:2010.10999*.

A Additional Index Size Analysis

This section contains additional index size analysis analogous to Figure 4 on Trivia-Open test data shown in Figure 5.
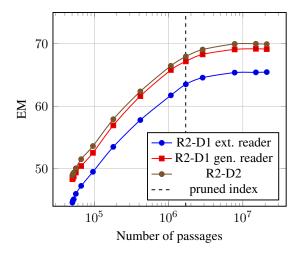

Figure 5: Index size analysis on TQ-Open test data.

B Additional Accuracy Analysis

Analysis of Accuracy@K on EfficientQA data is shown in Figure 7, on NQ-open validation data in Figure 6 and on TriviaQA validation data in Figure 8 and test data in Figure 9. The pruned version of our pipeline with reranker (reranked-pruned) performs better than the full version only with retriever (retrieved-full):

- on NQ-open (dev) up to $K = 32$ paragraphs,
- on NQ-open (test) up to $K = 26$ paragraphs,
- on EfficientQA up to $K = 43$ paragraphs,
- on TriviaQA (dev) up to $K = 110$ paragraphs,
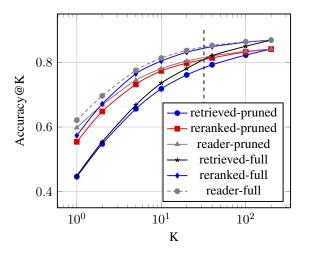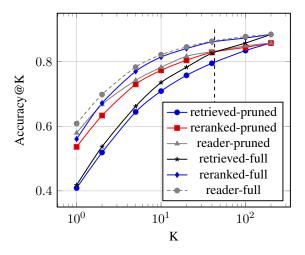- on TriviaQA (test) up to $K = 116$ paragraphs.

Figure 7: Analysis of Accuracy@K on EfficientQA.

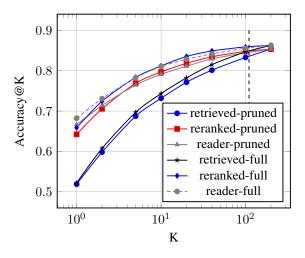Figure 8: Analysis of Accuracy@K on TriviaQA (dev).

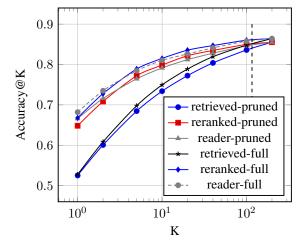Figure 6: Analysis of Accuracy@K on NQ-Open (dev).

Figure 9: Analysis of Accuracy@K on TriviaQA (test).

C Additional Component Fusion Analysis

This section includes results analogical to Tables 4, 5 on validation data of NQ-Open (Tables 6, 7), EfficientQA (Tables 8, 9) and TQ-Open (Tables 10, 11, 12, 13).

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	48.38	48.94	48.85	49.14
$\{g\}$	49.99	50.49	50.35	50.47
$\{e, g\}$	51.79	**51.97**	51.82	51.80

Table 6: Score aggregation – NQ-Open (dev).

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	50.65	51.24	51.01	51.17
$\{g\}$	50.36	50.91	50.68	50.90
$\{e, g\}$	52.24	**52.29**	52.27	52.07

Table 7: Binary decision – NQ-Open (dev).

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	47.56	48.33	48.89	48.72
$\{g\}$	49.11	49.56	50.22	50.11
$\{e, g\}$	50.78	51.67	50.89	**52.00**

Table 8: Score aggregation – EfficientQA.

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	48.33	50.06	49.39	49.67
$\{g\}$	48.94	49.50	50.06	49.72
$\{e, g\}$	50.78	51.83	50.94	**52.22**

Table 9: Binary decision – EfficientQA.

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	65.07	65.21	65.16	65.24
$\{g\}$	67.68	67.72	67.73	67.76
$\{e, g\}$	68.13	**68.19**	68.17	68.12

Table 10: Score aggregation – TQ-Open (dev).

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	69.03	69.03	69.01	68.99
$\{g\}$	69.54	69.46	69.62	69.70
$\{e, g\}$	69.77	**69.79**	69.67	69.61

Table 11: Binary decision – TQ-Open (dev).

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	65.54	65.64	65.60	65.61
$\{g\}$	68.25	68.17	68.21	68.26
$\{e, g\}$	68.45	68.57	**68.66**	**68.66**

Table 12: Score aggregation – TQ-Open (test).

$\boldsymbol{P}_*$	$\emptyset$	$\{r\}$	$\{rr\}$	$\{r, rr\}$
$\{e\}$	69.34	69.28	69.23	69.26
$\{g\}$	69.76	69.71	69.65	69.77
$\{e, g\}$	69.80	69.89	69.88	**69.94**

Table 13: Binary decision – TQ-Open (test).

D Data Pre-processing

This section describes how the training datasets for reranker and extractive reader are filtered, and how the distant supervision labeling is generated. Note not each example contains golden passage, as not each example can be mapped to the used dump of Wikipedia. We use the same golden passage mapping as Karpukhin et al. (2020).

For passage reranking, the input must contain at least one positive example. We meet this condition either by adding a golden passage or searching for the passage with an answer in the top-400 results retrieved by DPR. In detail about the search, first the Simple tokenizer proposed in DrQA[13] tokenizes each passage and golden answer. The positive example is the best-scored tokenized passage that contains an exact match with one of the tokenized answers. Note the search proceeds in the same way as in DPR[14] implementation.

The extractive reader is trained only on samples which contain exact match to at least one of the annotated answers in the top-1 passage, or golden passage if it is available. The exact match is performed on the subword token level (i.e. in ELEC-TRA's tokenization).

Next, the span annotations are extracted from the passages at the reader's input. Note each sample may contain multiple answers. The annotations for each answer in each sample are obtained differently in retrieved passages and in the golden passage. For retrieved passages, we search for the answer's exact matches in passages, and use each match as target annotation. For golden passage, we also search for the answer's exact matches in it. If there is none, the answer is soft-matched with single sub-sequence of golden passage, which yields

---

[13]https://github.com/facebookresearch/DrQA
[14]https://github.com/facebookresearch/DPR

highest non-zero F1 score. The F1 soft-match is also performed on the subword token level. Therefore answers with zero highest F1 soft-match with golden passage and no exact match in any of the reader's input passages are discarded.

Because the brute-force computation of a span with the greatest nonzero F1 score is potentially very demanding, we found the length limit for spans that are worth searching (see Theorem D.2).

To compare brute-force with upper bound implementation, we run an experiment on 16,741 passages (retrieved for NQ-Open dev). The average time per passage for brute-force was 121 ms and only 9 ms for implementation with an upper bound.

The soft match is described in Algorithm 1. It assumes that there is no exact match.

Algorithm 1 Soft match

Require: set of spans S and answer span a
 1: **function** SOFTMATCH(S, a)
 2:     actSize $\leftarrow 1$
 3:     lenLimit $\leftarrow 2$
 4:     bestSpan $\leftarrow$ None
 5:     bestScore $\leftarrow 0$
 6:     **while** actSize $<$ lenLimit **do**
 7:         **for all** $t \in S$ of size actSize **do**
 8:             score $\leftarrow \mathrm{F1}(t, a)$
 9:             **if** score $>$ bestScore **then**
10:                 bestSpan $\leftarrow t$
11:                 bestScore $\leftarrow$ score
12:                 lenLimit $\leftarrow |a|\frac{|t|+|a|-s_{ta}}{s_{ta}}$
13:         actSize $\leftarrow$ actSize $+ 1$
14:     **return** bestSpan

Lemma D.1. *Let t and a be non-empty spans and $0 < s_{ta} \leq |a|$ number of shared tokens for them. Then*[15]

$$|t| \leq |a|\frac{|t|+|a|-s_{ta}}{s_{ta}} . \qquad (13)$$

Proof. To prove it by contradiction assume that

$$|t| > |a|\frac{|t|+|a|-s_{ta}}{s_{ta}} , \qquad (14)$$

then

$$s_{ta}|t| > |a||t| + |a||a| - |a|s_{ta} , \qquad (15)$$

and also $0 < s_{ta} \leq |a|$, thus $|a||a| - |a|s_{ta} \geq 0$. Therefore even if we assume that

---
[15]|x| symbolises number of tokens in span x

$|a||a| - |a|s_{ta} = 0$. We get

$$s_{ta}|t| > |a||t|$$
$$s_{ta} > |a| , \qquad (16)$$

which is in contradiction with $0 < s_{ta} \leq |a|$. $\square$

Theorem D.2. *Let S be a set of non-empty spans, a an non-empty answer span, t non-empty trial span, $0 < s_{ta} \leq |a|$ is number of shared tokens for t and a, and $S_b = \{z | z \in S \wedge |z| \geq |a|\frac{|t|+|a|-s_{ta}}{s_{ta}}\}$. Then the theorem states that*

$$\forall x \in S_b(\mathrm{F1}(x, a) \leq \mathrm{F1}(t, a)) . \qquad (17)$$

Proof. To prove it by contradiction assume that

$$\exists x \in S_b(\mathrm{F1}(x, a) > \mathrm{F1}(t, a)) . \qquad (18)$$

F1 score can be expressed as:

$$\mathrm{F1}(b, c) = \frac{2s_{bc}}{|b| + |c|} , \qquad (19)$$

thus

$$\frac{2s_{xa}}{|x| + |a|} > \frac{2s_{ta}}{|t| + |a|} . \qquad (20)$$

From Lemma D.1 $|t| \leq |x|$. Therefore $s_{ta} < s_{xa}$, to satisfy the inequality (in equation 20), and we know that $0 < s_{xa} \leq |a|$. So let the $s_{xa} = |a|$ (the maximum) then

$$\frac{2|a|}{|x| + |a|} > \frac{2s_{ta}}{|t| + |a|}$$
$$|a|(|t| + |a|) > s_{ta}|x| + s_{ta}|a| \qquad (21)$$
$$|x| < |a|\frac{|t| + |a| - s_{ta}}{s_{ta}} ,$$

which is in contradiction with $x \in S_b$. $\square$

E Softmax Notation

Usually, softmax function $\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$ is defined as:

$$\sigma(v)_i = \frac{e^{v_i}}{\sum_{j=1}^{K} e^{v_j}}. \qquad (22)$$

However, some parts of this work used variant of softmax that is defined as follows:

$$\operatorname*{softmax}_{x \in D} \big(f(x)\big)_y = \frac{e^{f(y)}}{\sum_{x \in D} e^{f(x)}} , \qquad (23)$$

where D is the input set, $f : D \rightarrow \mathbb{R}$, $y \in D$.

F Decoding the Distributions from the Extractive Reader

We analyzed the subsets of joint probability space over spans obtained via multiplication of distributions as explained in section 3.2 in Table 14. The factors of this space are the distribution given by the outer product of independent probability distributions $\boldsymbol{P}_{start}(.)\boldsymbol{P}_{end}(.)^{\top}$ denoted as I, joint probability distribution $\boldsymbol{P}_{joint}(.)$ denoted as J, and passage distribution $\boldsymbol{P}_{passage}(.)$ denoted as C.

Factorization	NQ-dev	NQ-test	EfficientQA
I	48.32	50.58	47.33
J	48.53	**51.25**	**47.83**
I+J	**48.57**	50.83	**47.83**
I+C	48.22	50.55	47.22
J+C	48.49	51.11	47.56
I+J+C	48.50	50.86	47.67

Table 14: The results of extractive reader with different types of distribution used for decoding. See text for details.

G Passage Reranker Revision

In previous version of this work we used a Longformer encoder (Beltagy et al., 2020) with concatenated passages at it's input to benefit from the early fusion between passages. Therefore each passage was scored not only according to the question but also according to other passages. However, we did not observe any significant benefits when we used the Longformer setup over a RoBERTa which scores each passage independently (see Table 15).

H LDA Analysis of Pruned Passages

We apply LDA (Blei et al., 2003) with 100 topics to random subset of 1M passages from knowledge base. Then we apply T-SNE (Van der Maaten and Hinton, 2008) and project LDA vectors into 2 dimensions. The results are shown in Figure 10. The pruned passages seem to be evenly distributed through the topics.

Index	Readers	Fusion	NQ-Open (dev)			NQ-Open (test)			EfficientQA		
			Long.	RoB.	Δ	Long.	RoB.	Δ	Long.	RoB.	Δ
	ext	-	46.88	46.64	-0.24	48.81	48.92	0.11	45.22	45.06	-0.16
	gen	-	47.12	47.11	-0.01	48.39	48.31	-0.08	45.56	45.22	-0.34
1.7M	ext+gen	naive	47.71	47.78	0.07	50.55	50.33	-0.22	46.94	46.78	-0.16
	ext+gen	aggr	50.17	49.89	-0.28	52.11	52.38	0.27	49.06	49.44	0.38
	ext+gen	aggr+bd	50.50	50.25	-0.25	51.99	52.58	0.59	48.61	49.22	0.61
	ext	-	48.50	48.38	-0.12	50.86	50.72	-0.14	47.67	47.56	-0.11
	gen	-	49.34	49.40	0.06	51.50	50.69	-0.81	47.33	47.33	0.00
21M	ext+gen	naive	49.91	49.99	0.08	53.43	52.44	-0.99	49.06	49.11	0.05
	ext+gen	aggr	52.05	51.80	-0.25	54.96	54.90	-0.06	51.56	52.00	0.44
	ext+gen	aggr+bd	52.36	52.07	-0.29	55.01	54.99	-0.02	51.06	52.22	1.16

Table 15: Exact match comparison of Longformer (Long.) and RoBERTa (RoB.) based passage reranker.
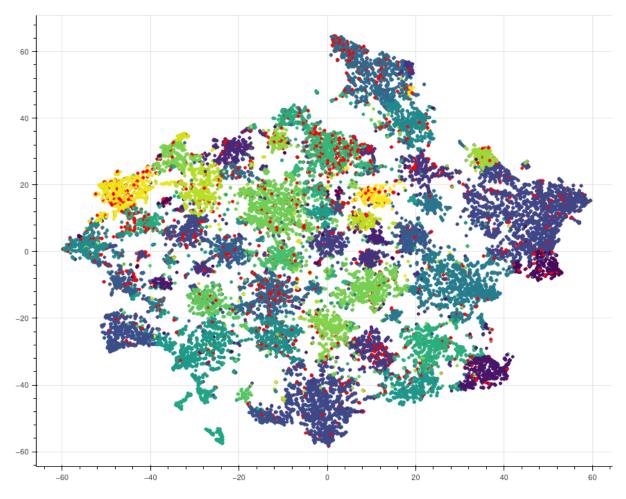

Figure 10: T-SNE plot of Latent dirichlet allocation (LDA) trained over 1M passages of Wikipedia. Red points are pruned passages from the 1.7M set. Other colors mark passages by their most dominant topic found by LDA. We remove passages with maximum topic weight lesser than 0.2 to clusterize the plot data.