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, surpassing the 50% exact match on NaturalQuestions and EfficientQA datasets. Specifically, it proposes the novel R2-D2 (RANK TWICE, READ TWICE) pipeline composed of retriever, 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 3% EM accuracy.\(^1\)

1 Introduction

Recent advances in neural passage retrieval (Karpukhin et al., 2020; Izacard and Grave, 2020a; Khattab et al., 2020; Guu et al., 2020) 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 dataset NaturalQuestions\(^2\). 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, reranker and 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)\(^3\).

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/. Our code and preprocessed data will be available online.

\(^2\)We will extend the pool of datasets in the next revision.

\(^3\)Leaderboard available at https://efficientqa.github.io/.
Wikipedia, we propose an apriori relevance classifier (we call \textit{pruner}) into relevance class $r$ that models the Bernoulli distribution $P(r|p_i)$. 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 \texttt{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 $P(r|p_i)$ for each passage $p_i$ in the corpus. We keep only passages $p_i$ that satisfy the threshold constraint $P(r|p_i) > \tau$; $\tau \in (0, 1)$.

### 3 Open-QA Pipeline

To estimate the impact of corpus pruning on various open-QA components, we propose a pipelined system R2-D2 (\textit{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 $q \in Q$ from the set of all possible questions $Q$ and the corpus $C = \{p_1, p_2, ..., p_n\}$ composed of passages $p_i$, the retriever learns a ranking function $\text{rank} : Q \times C \rightarrow \mathbb{R}$ that assigns a score to each passage. Note each passage contains its passage title.

Taking a top-$K$ scoring passages $C_r$, reranker again re-scores $C_r$ scoring passages by learning a reranking function $\text{rerank} : Q \times C_r \rightarrow \mathbb{R}$. 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 multiple passages.

Next, we experiment with two readers: the extractive reader reads top-$V$ passages $C_{rr}$ independently and assigns probability $P_e(a_e|q, C_{rr})$ to each span $a_e$ in the passages (see subsection 3.2). The FiD generative reader reads top-$V_2$ passages $C'_{rr}$ and generates an answer from probability space $P_g(a_g|q, C'_{rr})$ via greedy search.

Finally, R2-D2 aggregates the outputs from all components using two fusions (described in subsection 3.3).

#### 3.1 Passage Reranker

Proposed passage reranker is based on a transformer encoder with linear attention scaling, so that it can efficiently encode long sequences. For instance, one can choose Longformer (Beltagy et al., 2020).
(Zaheer et al., 2021).

The passages from retriever $p'_i \in C_r$ are shuffled and split into $N$ parts so the encoder received a concatenation of question $q \in Q$, special tokens and passages up to maximum length. There are $N$ inputs that look like

$$[CLS] \; q \; [SEP] \; p'_1 \; p'_2 \; \ldots \; p'_n \; [EOS].$$

Each passage consists of a title and context that start with a special token (<title> or <context>) and both are concatenated together.

The encoder computes a contextual representation for each of the $N$ input sequences $H_1, H_2, \ldots, H_N \in \mathbb{R}^{d \times l}$, where $l$ denotes maximum sequence length and $d$ dimension.

Vector representation $\tilde{h}_i \in \mathbb{R}^{2d}$ of passage $p'_i$ is the concatenation of vectors $h_i^{<\text{title}>}$ and $h_i^{<\text{context}>}$:

$$\tilde{h}_i = [h_i^{<\text{title}>}; h_i^{<\text{context}>}]. \quad (1)$$

These vectors corresponding to <title> and <context> token of that passage and are rows from matrix $H_m$, where $m$ is the input sequence containing the passage. Note that $[\cdot ; \cdot ; \cdot]$ notation denotes concatenation.

Now we are able to define reranking function to re-score passage $p'_i$ as

$$\text{rerank}(q, p'_i) = w^\top \tilde{h}_i \quad (2)$$

where $w \in \mathbb{R}^{2d}$ is a trainable vector. Finally, we define the following formula

$$P_{rr}(p'_i | q, C_r) = \text{softmax} (\text{rerank}(q, p))_{p \in C_r} \quad (3)$$

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

During the training, each input sequence contains exactly one ground truth passage and hard negatives passages up to maximum length. Hard negatives are uniformly sampled from $C_r$, which do not contain answer to the question. The loss function of the passage reranker was the cross entropy.

### 3.2 Extractive Reader

Extractive reader estimates the probability $P_e(a_e | q, C_{rr})$. It is the probability of a span $a_e$ from top-V passage $p \in C_{rr}$ being an answer to a question $q$. We decompose the $P_e(a_e | q, C_{rr})$ 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).

These probabilities are defined as:

$$P_s(* | q, C_{rr}) = \text{softmax}(s_i), \quad (4)$$

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

$$s_{\text{start}} = \text{En}(p, q)[s]^\top w_{\text{start}} \quad (5)$$

$$s_{\text{end}} = \text{En}(p, q)[e]^\top w_{\text{end}} \quad (6)$$

$$s_{\text{joint}} = (W_j \text{En}(p, q)[s] + b_j)^\top \text{En}(p, q)[e] \quad (7)$$

$$s_{\text{passage}} = \text{En}(p, q)[\text{CLS}]^\top w_p. \quad (8)$$

Where $w_s, b_j \in \mathbb{R}^h$, $\text{En}(p, q)[\cdot] \in \mathbb{R}^h$, and $W_j \in \mathbb{R}^{h \times h}$ are all trainable. The encoder (En) used in our experiments was the ELECTRA (Clark et al., 2020). CLS is a special token added at the start of an input sequence (Devlin et al., 2019).

We omit the spans of a title 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:

$$- \log \sum_{s \in \text{starts}(C_{rr})} P_{\text{start}}(s|q, C_{rr})$$

$$- \log \sum_{e \in \text{ends}(C_{rr})} P_{\text{end}}(e|q, C_{rr})$$

$$- \log \sum_{j \in \text{boundaries}(C_{rr})} P_{\text{joint}}(j|q, C_{rr}) \quad (9)$$

$$- \log \sum_{p \in C_{rr}} P_{\text{passage}}(p|q, C_{rr}).$$

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 extractive reader are aggregated. Formally, assume the \(A_q\) is the set of top-\(M\) answer spans from \(P_e(a|q, C_{rr})\) for question \(q\). The generative model performs the answer reranking evaluating the log-probability of the answer spans

\[
\{ \log P_g(a|q, C_{rr}^\prime) : a \in A_q \}. \tag{10}
\]

Next a logistic regression loss (12) is minimized to perform score aggregation. It combines the scores across the R2-D2 components to maximize the correct answer span probability over dataset \(D\).

This dataset is composed of the TOP-\(M\) outputs of the extractive reader with the correct answer.

\[
x(a) = [P_e(a) P_g(a) P_r(p_a) P_{rr}(p_a)]
- \sum_{(A_q, gt) \in D} \frac{1} {a \in A_q} \max \left( w^\top \log x(a) + b \right)_{gt} \tag{12}
\]

Here \(p_a\) denotes the passage containing the answer span \(a\), \(A_q\) 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 collection \(C_{rr}\), 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 language model. Given that \(s_{agg}(q) = \max_{a \in A_q} \frac {w^\top x(a) + b} {a^*}\) is the best span score and \(s_{agg}^*(q) = \log P_g(a^*|q, C_{rr}^\prime)\) is the log-probability of the answer \(a^*\) obtained via greedy decoding for question \(q\), a classifier is trained via binary cross-entropy (BCE) to do the binary decision

\[
\sum_{(e,t) \in D} BCE(\log(w^\top [s_{agg}(e); s_{agg}^*(e)] + b)). \tag{13}
\]

Here, the training dataset \(D\) 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 reranker, 48GB GPU for the generative reader, and 16x 32GB GPUs to train the extractive reader with the large batch size. 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)\(^4\).

**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)\(^5\).

4.1 Datasets and Data Pre-processing

We evaluate our models on two datasets. Their statistics are available in Table 1. To train the reranker and extractive reader, we filter out examples, which:

1. do not contain ground-truth passage annotation in data mapped to Wikipedia, the same way as in Karpukhin et al. (2020),

2. and are not matched via our F1 heuristic (described in Appendix C).

**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.

**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)).

\(^4\)https://cutt.ly/rrZNl4r

\(^5\)https://cutt.ly/0luNh4
Table 1: Dataset statistics. For train, the left column is full dataset, the right is filtered only to relevant examples. The golden set is a dataset used for pruner training.

| Dataset      | Train   | Dev    | Test   |
|--------------|---------|--------|--------|
| NQ-Open      | 79,168  | 6,755  | 3,610  |
| EfficientQA  | -       | -      | -      | 1,800  |
| NQ-Golden    | 176,628 | 4,332  | 8,698  |

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 our binary classifier. To train the system, we create training set with 2 negative samples per positive passage from filtered examples. We split the dataset’s development set into development and test set in a 1 : 2 ratio. We keep only one negative passage per filtered sample for development and test so that datasets are balanced. We further refer to these datasets as Golden. The system is trained via cross-entropy in 2 epochs using batch size 12 and learning rate $3 \times 10^{-5}$ linearly decreasing to 0. The $\tau$ threshold is tuned so that we pool the number of passages to fit the 6GiB limit. We use pruner to extract the top 1,7M most relevant passages, and we combine these with missing golden passages from the training data, obtaining 1,702,133 passages in total for NQ-open.

Retriever. We use BERT-based DPR from the official checkpoint. Each passage is represented via 768-dimensional embedding. 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 $C_r$ to reranker.

Passage reranker. We use the Longformer-base encoder, which was trained on documents of maximum length 4,096. Global attention was applied to question tokens following the Beltagy et al. (2020). We use a linear scheduler with 0.1 warmup proportion, the number of epochs is 5 and the model is validate every 40,000 optimization steps. The initial learning rate is $2 \times 10^{-4}$ and batch size equals to 8.

Extractive reader. The extractive reader is based on pre-trained ELECTRA-large. During the training phase, all spans from all $p \in C_r$ that match 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 128 top passages during the training phase and when it is used without the reranker. If the reranker is used, 24 top passages are taken instead.

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 $2 \cdot 10^{-5}$.

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. In training, the golden passage always comes first. Due to the large memory 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 compression to reduce the size of the models and the corpus. To fit the 6GiB limit, we use 100MB CentOS8 docker image 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 3 times fewer passages at its input, as in Izacard and Grave (2020b). In contrast, our ELECTRA based extractive reader R1-D1 (Extractive) shows large gains compared to extractive state-of-the-art, while having the same retriever as DPR.

Note that we use the retriever output directly.

Matching strategies are described in Appendix C.

https://launchpad.net/ubuntu/+source/zip

nvidia/cuda:10.2-base-centos8
| Method                                | NQ   | #θ   |
|--------------------------------------|------|------|
| BM25+BERT (Mao et al., 2020)         | 37.7 | 110M |
| Hard EM (Min et al., 2019a)          | 28.1 | 110M |
| Path Retriever (Asai et al., 2019)   | 32.6 | 447M |
| Graph Retriever (Min et al., 2019b)  | 34.5 | 110M |
| ORQA (Lee et al., 2019)              | 33.3 | 220M |
| REALM (Guu et al., 2020)             | 40.4 | 660M |
| ProQA (Xiong et al., 2020)           | 34.3 | 220M |
| DPR (Karpukhin et al., 2020)         | 41.5 | 220M |
| DPR-subset* (Min et al., 2021)       | 34.8 | 220M |
| RDR (Yang and Seo, 2020)             | 42.1 | 110M |
| GAR+DPR (Mao et al., 2020)           | 43.8 | 626M |
| ColBERT (large) (Khattab et al., 2020)| 48.2 | 440M |
| BM25+SSG (Mao et al., 2020)          | 35.3 | 406M |
| T51.1+SSM (Roberts et al., 2020)     | 35.2 | 11B  |
| RAG (Lewis et al., 2020)             | 44.5 | 516M |
| DPR+SGG (Min et al., 2020)           | 42.2 | 516M |
| FiD-base (Izacard and Grave, 2020b) | 48.2 | 333M |
| FiD-large (Izacard and Grave, 2020b) | 51.4 | 848M |
| FiD-large+* (Izacard et al., 2020)   | 53.6 | 848M |
| FiD-large++ (Izacard et al., 2020)   | 54.7 | 848M |
| RIDER (GAR+DPR) (Mao et al., 2021)   | 48.3 | 626M |

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.

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 batch size (128) 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.

**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 $K = 36$ paragraphs. We observe the similar trend on other datasets (see Appendix B).

**Pruner.** Our simple pruning approach achieved 90.63% accuracy on NQ-Golden test data. Interestingly, it still missed 2,133/40,670 (5.24%) golden passages from the 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 shown in Figure 3. The total uncompressed size of an image is 81.01GiB while the size of the pruned image is 5.96 GiB (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 available 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).
Table 3: Ablation study. The $\Delta$ column shows the exact match difference caused by pruning. Namely, performing a naive answer re-ranking by generative reader means the system chooses the most probable answer span according to generative reader log-probabilities as shown in equation (10). Analogously, the aggr fusion denotes that the system chooses the most probable answer span according to aggregated scores, as in equation (12). Finally, the aggr+bd fusion denotes the binary decision, as shown in equation (13). Interestingly, the results on NQ-Open (1.7M) suggest the reranker might not always improve the performance significantly. Additionally, the results suggest the binary decision does not improve results significantly, and more advanced technique for combining the extractive and the abstractive answer is needed. However, this inferior performance of the binary decision might not be always the case (see further text).

Table 4: Score aggregation.

Table 5: Binary decision.

Component fusion. Finally, we analyze the 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. Table 4 shows all relevant combinations of ranker $r$, reranker $rr$, extractive reader $e$ and generative reader $g$ probabilities used in score aggregation. Interestingly, the scores from the retriever and the reranker help significantly only if generative answer reranking is not used. The impact of adding a binary decision after the score aggregation is shown in Table 5. Interestingly, as in the previous finding, the binary decision component significantly improves the performance only without reranked answer scores (first row in both tables). However, fusing the generative and extractive reader via binary decision performs significantly worse than fusing both readers together with score aggregation (first row in Table 5 vs. last row in Table 4).

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 passage, 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,
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 outook 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 (2020b) 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 does not require any training.

The above works score each document independently or estimate the more relevant from two documents. However, our work is different in the early passage concatenation. Therefore each document is scored not only according to the question but also according to other documents.

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 Discussion

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 ap-
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). Furthermore, we showed all system components play an important role in our pipeline.

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. Please note this pre-print is a work-in-progress and we plan to cover more in-depth analyses in future revisions.\(^\text{11}\)

Acknowledgments

We would like to thank Jan Doležal for implementing an R2-D2 demo.

This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project „IT4Innovations National Supercomputing Center – LM2018140“.

References

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

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

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In \textit{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 \textit{Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics}, pages 5657–5667.

Christopher Clark and Matt Gardner. 2018. \textit{Simple and effective multi-paragraph reading comprehension}. In \textit{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. \textit{ELECTRA: Pre-training text encoders as discriminators rather than generators}. In \textit{International Conference on Learning Representations}.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. \textit{BERT: Pre-training of deep bidirectional transformers for language understanding}. In \textit{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)}, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

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

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

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

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

\(^\text{11}\)If you would like to propose an interesting direction towards analyzing our system in future revisions, do not hesitate to write us an email.
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).

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, Lingqing 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.

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

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, Zhiyi 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, Natalie 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.*

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021.
Big Bird: Transformers for longer sequences.
arXiv:2007.14062.
A Additional Component Fusion Analysis

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

B Additional Accuracy Analysis

Analysis of Accuracy@K on EfficientQA data is shown in Figure 4. The pruned version of our pipeline with reranker (reranked-pruned) performs better than the full version only with retriever (retrieved-full) up to $K = 53$ paragraphs.

C F1 Answer Matching

The extractive reader uses distant supervision. The target answer annotations were obtained by searching a match in the passage context with an answer on the subword token level. We distinguish two different match strategies:

- **exact match**
  - A sequence of subword tokens of an answer and a context span is the same.

- **soft match**
  - Span with the biggest nonzero F1 match with the given answer span is selected.

Which match is used depends on whether we know which passage in a batch is the ground truth passage, as not every question is mapped to the knowledge-base. The strategies are as follows:

- **ground truth passage is known**
  - only the exact match is used

- **unknown ground truth passage**
  - We search every passage context for an exact match answer span, and if there is not any, we select as the answer span the span with the greatest nonzero F1 score among all spans in all passages.

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.

Let us say that one wants to ask how much is it still profitable to increase the length of investigated spans if spans of length $l$ have been already investigated and the maximal F1 for them is known.

---

| $P_*$ | $\emptyset$ | $\{r\}$ | $\{rr\}$ | $\{r, rr\}$ |
|-------|-----------|---------|----------|-------------|
| $\{e\}$ | 48.50 | 49.06 | 48.77 | 49.27 |
| $\{g\}$ | 49.91 | 50.51 | 50.26 | 50.53 |
| $\{e, g\}$ | 51.95 | 52.18 | 52.02 | **52.10** |

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

| $P_*$ | $\emptyset$ | $\{r\}$ | $\{rr\}$ | $\{r, rr\}$ |
|-------|-----------|---------|----------|-------------|
| $\{e\}$ | 50.71 | 51.20 | 50.89 | 51.31 |
| $\{g\}$ | 50.18 | 50.81 | 50.54 | 50.84 |
| $\{e, g\}$ | 52.31 | **52.46** | 52.35 | 52.40 |

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

| $P_*$ | $\emptyset$ | $\{r\}$ | $\{rr\}$ | $\{r, rr\}$ |
|-------|-----------|---------|----------|-------------|
| $\{e\}$ | 47.67 | 48.67 | 48.83 | 48.83 |
| $\{g\}$ | 49.06 | 49.78 | 50.33 | 51.28 |
| $\{e, g\}$ | 50.89 | **51.72** | 51.28 | 51.56 |

Table 8: Score aggregation – EfficientQA

| $P_*$ | $\emptyset$ | $\{r\}$ | $\{rr\}$ | $\{r, rr\}$ |
|-------|-----------|---------|----------|-------------|
| $\{e\}$ | 48.83 | 50.44 | 49.50 | 50.11 |
| $\{g\}$ | 48.83 | 49.56 | 50.22 | 51.11 |
| $\{e, g\}$ | 50.83 | **51.67** | 51.11 | 51.06 |

Table 9: Binary decision – EfficientQA
Because the F1 score can be expressed as:

$$F1 = \frac{2s}{t + a},$$  \hspace{1cm} (14)

where $s$ is the number of shared tokens, $t$ is the number of trial span tokens, and $a$ is the number of answer span tokens. It can be seen that the F1 increases when the $s$ increases or $t$ decreases ($a$ is fixed). If the $l = 1$ it could not be further decreased. When $l > 1$, we already know the maximal F1 span for all spans with size $<= l$. So we can only make an optimistic assumption and state that we will increase the number of shared tokens by increasing the trial span size. The maximum number of shared tokens is $s = a$. Thus we get this F1 upper bound:

$$\frac{2a}{xa + a},$$  \hspace{1cm} (15)

where we expressed the $t$ as $t = xa$. To get the parameter $x$, let us define the following inequality that represents the profit condition:

$$\frac{2s_k}{l + a} < \frac{2a}{xa + a},$$  \hspace{1cm} (16)

where the $s_k$ is the already known number of shared tokens of trial span with maximal F1 match so far, and $l$ is its length.

Thus we can express the $x$:

$$\frac{2s_k}{l + a} < \frac{2a}{xa + a}$$

$$s_k < \frac{1}{x + 1}$$

$$x < \frac{l + a - s_k}{s_k}$$  \hspace{1cm} (17)

Therefore we do not need to investigate trial spans with size

$$a \frac{l + a - s_k}{s_k}$$

or greater, if we know that the best trial span has size $l$ and shares $s_k$ tokens.

It is an iterative process, and when we find a trial span with a greater F1 score, it must be recalculated.

Also, it is worth mentioning that when the $s_k = 0$, we can omit the search because we did not find any common token for any span at size $l$.

### D Softmax notation

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

$$\sigma(v)_i = \frac{e^{v_i}}{\sum_{j=1}^{K} e^{v_j}},$$  \hspace{1cm} (19)

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

$$\text{softmax}_{x \in D}(f(x))_y = \frac{e^{f(y)}}{\sum_{x \in D} e^{f(x)}},$$  \hspace{1cm} (20)

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