PAQ: 65 Million Probably-Asked Questions and What You Can Do With Them

Patrick Lewis[†‡] **Yuxiang Wu**[‡] **Linqing Liu**[‡] **Pasquale Minervini**[‡] **Heinrich Küttler**[†]
Aleksandra Piktus[†] **Pontus Stenetorp**[‡] **Sebastian Riedel**[†‡]

[†]Facebook AI Research [‡]University College London, UK
{plewis,hnr,piktus,sriedel}@fb.com
{yuxiang.wu,linqing.liu,p.minervini,p.stenetorp}@cs.ucl.ac.uk

Abstract

Open-domain Question Answering models that directly leverage question-answer (QA) pairs, such as closed-book QA (CBQA) models and QA-pair retrievers, show promise in terms of speed and memory compared with conventional models which retrieve and read from text corpora. QA-pair retrievers also offer interpretable answers, a high degree of control, and are trivial to update at test time with new knowledge. However, these models fall short of the accuracy of retrieve-and-read systems, as substantially less knowledge is covered by the available QA-pairs relative to text corpora like Wikipedia. To facilitate improved QA-pair models, we introduce *Probably Asked Questions* (PAQ), a very large resource of 65M automatically generated QA-pairs. We introduce a new QA-pair retriever, RePAQ, to complement PAQ. We find that PAQ *preempts* and *caches* test questions, enabling RePAQ to match the accuracy of recent retrieve-and-read models, whilst being significantly faster. Using PAQ, we train CBQA models which outperform comparable baselines by 5%, but trail RePAQ by over 15%, indicating the effectiveness of explicit retrieval. RePAQ can be configured for size (under 500MB) or speed (over 1K questions per second) while retaining high accuracy. Lastly, we demonstrate RePAQ's strength at *selective QA*, abstaining from answering when it is likely to be incorrect. This enables RePAQ to ''back-off'' to a more expensive state-of-the-art model, leading to a combined system which is both more accurate and 2x faster than the state-of-the-art model alone.

1 Introduction

Open-domain QA (ODQA) systems usually have access to a background corpus that can be used to answer questions. Models that explicitly exploit this corpus are commonly referred to as *Open-book* models (Roberts et al., 2020). They typically index the whole corpus, and then *retrieve-and-read* documents to answer questions (Voorhees and Harman, 1999; Chen et al., 2017, inter alia).

A second class of models, *closed-book* question answering (CBQA) models, has recently been proposed. They learn to directly map questions to answers from training question-answer (QA) pairs without access to a background corpus (Roberts et al., 2020; Ye et al., 2021). These models usually take the form of pretrained seq2seq models such as T5 (Raffel et al., 2020) or BART (Lewis et al., 2020a), fine-tuned on QA-pairs. It has recently been shown that current closed-book models mostly memorize training QA-pairs, and can struggle to answer questions that do not overlap with training data (Lewis et al., 2021).

Models that explicitly retrieve (training) QA-pairs, rather than memorizing them in parameters, have been shown to perform competitively with CBQA models (Lewis et al., 2021; Xiao et al., 2021). These models have a number of useful properties, such as fast inference, interpretable outputs (by inspecting retrieved QA-pairs), and the ability to update the model's knowledge at test time by adding or removing QA-pairs.

However, CBQA and QA-pair retriever models are currently not competitive with retrieve-and-read systems in terms of accuracy, largely because the training QA-pairs they operate on cover substantially less knowledge than background corpora like Wikipedia. In this paper, we explore whether greatly expanding the coverage of QA-pairs enables CBQA and QA-pair retriever models which are competitive with retrieve-and-read models.

We present Probably Asked Questions (PAQ), a semi-structured Knowledge Base (KB) of 65M natural language QA-pairs, which models can memorise and/or learn to retrieve from. PAQ

Transactions of the Association for Computational Linguistics, vol. 9, pp. 1098–1115, 2021. https://doi.org/10.1162/tacl_a_00415
Action Editor: Michael Collins. Submission batch: 4/2021; Revision batch: 7/2021; Published 10/2021.

differs from traditional KBs in that questions and answers are stored in natural language, and that questions are generated such that they are likely to appear in ODQA datasets. PAQ is automatically constructed using a question generation model and Wikipedia. To ensure generated questions are not *only* answerable given the passage they are generated from, we use a *global filtering* post-processing step using a state-of-the-art ODQA system. This greatly reduces the amount of wrong/ambiguous questions compared to other approaches (Fang et al., 2020; Alberti et al., 2019), and is critical for high-accuracy, downstream QA.

To complement PAQ we develop RePAQ, an ODQA model based on question retrieval/matching models, using dense Maximum Inner Product Search-based retrieval, and optionally, re-ranking. We show that PAQ and RePAQ provide accurate ODQA predictions, at the level of relatively recent large-scale retrieve-and-read systems such as RAG (Lewis et al., 2020b) on NaturalQuestions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). PAQ instances are annotated with scores that reflect how likely we expect questions to appear, which can be used to control the memory footprint of RePAQ by pruning the KB accordingly. As a result, RePAQ becomes flexible, allowing us to configure QA systems with near state-of-the-art results, very small memory size, or inference speeds of over 1,000 questions per second.

We also show that PAQ is a useful source of training data for CBQA models. BART models trained on PAQ outperform standard data baselines by 5%. However, these models struggle to effectively memorize all the knowledge in PAQ, lagging behind RePAQ by 15%. This demonstrates the effectiveness of RePAQ at leveraging PAQ.

Finally, we show that as RePAQ's question matching score correlates well with QA accuracy, it effectively ''knows when it doesn't know'', allowing for *selective question answering* (Voorhees, 2002) where systems may abstain from answering. Although answer abstaining is important in its own right, it also enables an elegant ''back-off'' approach where we can defer to a more accurate but expensive QA system when the answer confidence is low. This allows us to make use of the best of both speed and accuracy.

In summary, we make the following contributions: i) introduce PAQ, 65M QA-pairs automatically generated from Wikipedia, and demonstrate the importance of global filtering for high quality; ii) introduce RePAQ, a QA system designed to utilize PAQ and demonstrate how it can be optimised for memory, speed, or accuracy; ii) investigate the utility of PAQ for CBQA models, improving by 5% but note significant headroom to RePAQ iv) demonstrate RePAQ's strength on selective QA, enabling us to combine RePAQ; and with a state-of-the-art QA model, making it both more accurate and 2x faster.[1]

2 Open-Domain Question Answering

ODQA is the task of answering natural language factoid questions from an open set of domains. A typical question might be ''when was the last year astronauts landed on the moon?'', with a target answer ''1972''. The goal of ODQA is to develop an answer function $m : Q \mapsto A$, where Q and A, respectively, are the sets of all possible questions and answers. We assume there is a distribution $P(q, a)$ of QA-pairs, defined over $Q \times A$. A good answer function will minimize the expected error over $P(q, a)$ with respect to some loss function, such as answer string match. In practice, we do not have access to $P(q, a)$, and instead rely on an empirical sample of QA-pairs $\mathcal{K}$ drawn from P, and measure the empirical loss of answer functions on $\mathcal{K}$. Our goal in this work is to implicitly model $P(q, a)$ in order to draw a large sample of QA-pairs, PAQ, which we can train on and/or retrieve from. A sufficiently-large drawn sample will overlap with $\mathcal{K}$, essentially *pre-empting* and *caching* questions that humans may ask at test-time. This allows us to shift computation from test-time to train-time compared to retrieve-and-read methods.

3 Generating Question-Answer Pairs

In this section, we describe the process for generating PAQ. Given a large background corpus C, our QA-pair generation process consists of the following components:

1. A *passage selection* model $p_s(c)$, to identify passages which humans are likely to ask questions about.

[1]Data, models, and code are available at `https://github.com/facebookresearch/PAQ`.

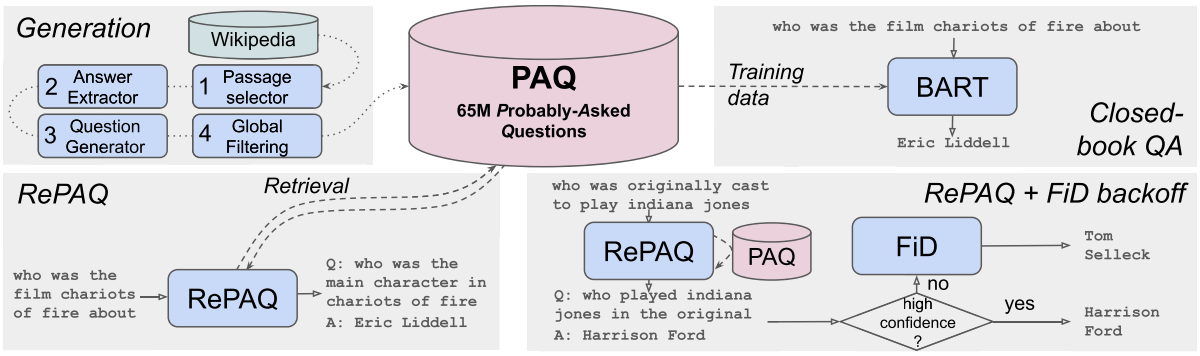

Figure 1: Top Left: Generation pipeline for QA-pairs in PAQ. Top Right: PAQ used as training data for CBQA models. Bottom Left: RePAQ retrieves similar QA-pairs to input questions from PAQ. Bottom right: RePAQ's confidence is predictive of accuracy. If confidence is low, we can defer to slower, more accurate systems, like FiD.

2. An *answer extraction* model $p_a(a|c)$, for identifying spans in a passage that are more likely to be answers to a question.

3. A *question generator* $p_q(q|a,c)$ that, given a passage and an answer, generates a question.

4. A *filtering* QA model $p_f(a|q,C)$ that generates an answer for a given question. If an answer generated by p_f does not match the answer a question was generated from, the question is discarded. This ensures generated questions are *consistent* (Alberti et al., 2019).

As shown in Figure 1, these models are applied sequentially to generate QA-pairs, similarly to *contextual* QA generation (Alberti et al., 2019; Lewis et al., 2019). First a passage c is selected with a high probability under p_s. Next, candidate answers a are extracted from c using p_a, and questions q are generated for each answer using p_q. Lastly, p_f generates a new answer a' for the question. If source answer a matches a', then (q,a) is deemed consistent and added to PAQ. The pipeline is based on Alberti et al. (2019), updated to take advantage of recent modeling advances. Passage selection and our filtering approach are novel contributions to the best of our knowledge, specifically designed for ODQA QA-pair generation. Each component is described in detail below.

3.1 Passage Selection, p_s

The passage selection model p_s is used to find passages that are likely to contain information that humans may ask about, and thus make good candidates to generate questions from. We learn p_s using a similar method to Karpukhin et al. (2020). Concretely, we assume access to a set of positive passages $C^+ \subset C$, obtained from answer-containing passages from ODQA train sets. As we do not have a set of labeled negatives, we sample negatives either randomly or using heuristics. We then maximize log-likelihood of positive passages relative to negatives. We implement p_s with RoBERTa (Liu et al., 2019) and obtain positive passages from Natural Questions (NQ, Kwiatkowski et al., 2019). We sample *easy negatives* at random from Wikipedia, and *hard negatives* from the same Wikipedia article as the positive passage. Easy negatives help the model to learn topics of interest, and hard negatives help to differentiate between interesting and non-interesting passages from the same article.

3.2 Answer Extraction, p_a

Given a passage, this component identifies spans that are likely to be answers to questions. We consider two alternatives: an off-the-shelf Named Entity Recognizer (NER) or training a BERT (Devlin et al., 2019) answer extraction model on NQ.

The NER answer extractor simply extracts all named entities from a passage.[2] The majority of questions in ODQA datasets consist of entity mentions (Kwiatkowski et al., 2019; Joshi et al., 2017), so this approach can achieve high answer coverage. However, as we extract all entity mentions in a passage, we may extract unsuitable mentions,

[2]We use the spaCy (Honnibal et al., 2020) NER system, trained on OntoNotes (Hovy et al., 2006).

or miss answers that do not conform to the NER system's annotation schema. The trained answer span extractor aims to address these issues.

BERT answer span extraction is typically performed by modelling answer start and end independently (Devlin et al., 2019). We instead follow the approach of Alberti et al. (2019), which breaks the conditional independence of answer spans by directly predicting $p_a(a|c) = p([a_{\text{start}}, a_{\text{end}}]|c)$. Our implementation first feeds a passage through BERT, before concatenating the start and end token representations of all possible spans of up to length 30, before passing them through an MLP to give $p_a(a|c)$. At generation time, we extract the top-K most probable spans from each passage.

3.3 Question Generation, p_q

Given a passage and an answer, this model generates likely questions with that answer. To indicate the answer and its occurrence in the passage, we prepend the answer to the passage and label the answer span with surrounding special tokens. We train on a combination of NQ, TriviaQA, and SQuAD, and perform standard fine-tuning of BART-base (Lewis et al., 2020a) to obtain p_q.

3.4 Filtering, p_f

The filtering model p_f improves the quality of generated questions, by ensuring that they are *consistent*—that the answer they were generated is likely to be a valid answer to the question. Previous work (Alberti et al., 2019; Fang et al., 2020) has used a machine reading comprehension (MRC) QA model for this purpose, $p_f(a|q, c)$, which produces an answer when supplied with a question *and* the passage it was generated from. We refer to this as *local filtering*. However, local filtering will not remove questions that are ambiguous (Min et al., 2020b), and can only be answered correctly with access to the source passage. Thus, we use an ODQA model for filtering, $p_f(a|q, C)$, supplied with only the generated question, and *not* the source passage. We refer to this as *global filtering*, and later show it is vital for strong downstream results. We use FiD-base with 50 passages, trained on NQ (Izacard and Grave, 2021).

4 Question Answering Using PAQ

We consider two uses of PAQ for building QA models. The first is to use PAQ as a source of training QA-pairs for CBQA models. The second treats PAQ as a KB, which models learn to directly retrieve from. These are related, as CBQA models have been shown to memorize the train data in their parameters, latently retrieving from them at test time (Lewis et al., 2021; Domingos, 2020).

4.1 PAQ for Closed-Book QA

We fine-tune BART-large (Lewis et al., 2020a) with QA-pairs from the concatenation of the training data and PAQ, using a similar training procedure to Roberts et al. (2020). We use a batch size of 512, and use validation Exact Match score for early stopping (Rajpurkar et al., 2018). Following recent best practices (Alberti et al., 2019; Yang et al., 2019), we then fine-tune on training QA-pairs only. We note that effective CBQA models must be able to understand the semantics of questions and how to generate answers, in addition to being able to store large numbers of facts in their parameters. This model thus represents a kind of combined *parametric* knowledgebase and retrieval system (Petroni et al., 2020). The model proposed in the next section, RePAQ, represents an explicit *non-parametric* instantiation of this idea.

4.2 RePAQ

RePAQ is a retrieval model that operates on KBs of QA-pairs, such as PAQ. RePAQ extends recently proposed nearest neighbor QA-pair retriever models (Lewis et al., 2021; Xiao et al., 2021). These models assume access to a KB of N QA-pairs $\mathcal{K} = \{(q_1, a_1) \ldots (q_N, a_N)\}$. These models provide an answer to a test question q by finding the most relevant QA-pair (q', a') in $\mathcal{K}$, using a scalable relevance function, then returning a' as the answer to q. This function could be implemented using standard information retrieval techniques, (e.g., TF-IDF) or learned from training data. RePAQ is learned from ODQA data and consists of a neural retriever, optionally followed by a reranker.

4.2.1 RePAQ Retriever

Our retriever adopts the dense Maximum Inner Product Search (MIPS) paradigm, which has recently been shown to obtain state-of-the-art results in a number of settings (Karpukhin et al., 2020; Lee et al., 2021, inter alia). Our goal is to

embed queries q and indexed items d into a representation space via embedding functions g_q and g_d, so that the inner product $g_q(q)^\top g_d(d)$ is maximized for items relevant to q. In our case, queries are questions and indexed items are QA-pairs (q', a'). We make our retriever symmetric by embedding q' rather than (q', a'). As such, *only one* embedding function g_q is required, which maps questions to embeddings. This applies a useful inductive bias, and we find that it aids stability during training.

Learning the embedding function g_q is complicated by the lack of labeled question pair paraphrases in ODQA datasets. We propose a latent variable approach similar to retrieval-augmented generation (RAG, Lewis et al., 2021),[3] where we index training QA-pairs rather than documents. For an input question q, the top K QA-pairs (q', a') are retrieved by a retriever $p_{\texttt{ret}}$ where $p_{\texttt{ret}}(q|q') \propto \exp(g_q(q)^\top g_q(q'))$. These are then fed into a seq2seq model $p_{\texttt{gen}}$ which generates an answer for each retrieved QA-pair, before a final answer is produced by marginalising,

$$p(a|q) = \sum_{(a',q') \in \text{top-}k\, p_{\texttt{ret}}(\cdot|q)} p_{\texttt{gen}}(a|q, q', a') p_{\texttt{ret}}(q'|q),$$

As $p_{\texttt{gen}}$ generates answers token-by-token, credit can be given for retrieving helpful QA-pairs that do not exactly match the target answer. For example, for the question ''when was the last time anyone was on the moon'' and target answer ''December 1972'', retrieving ''when was the last year astronauts landed on the moon'' with answer ''1972'' will help to generate the target answer, despite the answers having different granularity. After training, we discard $p_{\texttt{ret}}$,[4] retaining only the question embedder g. We implement $p_{\texttt{ret}}$ with ALBERT (Lan et al., 2020) with an output dimension of 768, and $\mathrm{p_{gen}}$ with BART-large (Lewis et al., 2020a). We train with 100 retrieved QA-pairs, and refresh the index every 5 training steps.

Once the embedder g_q is trained, we build a test-time QA system by embedding and indexing a QA KB such as PAQ. Answering is achieved by retrieving the most similar stored question, and returning its answer. The matched QA-pair can be displayed to the user, providing a mechanism for more interpretable answers than CBQA models and many retrieve-and-read generators which consume thousands of tokens to generate an answer. Efficient MIPS libraries such as FAISS (Johnson et al., 2019) enable RePAQ's retriever to answer 100s to 1,000s of questions per second (see Section 5.2.3). We use a KB for RePAQ consisting of train set QA-pairs and QA-pairs from PAQ.

4.2.2 RePAQ Reranker

Accuracy can be improved using a reranker on the top-K QA-pairs from the retriever. The reranker uses cross-encoding, and includes the retrieved answer in the scoring function for richer featurisation. The model is trained as a multi-class classifier, attempting to classify a QA-pair which answers a question correctly against K-1 retrieved QA-pairs which do not. For each QA-pair candidate, we concatenate the input question q with the QA-pair (q', a'), and feed it through ALBERT, and project the CLS representation to a logit score. The model produces a distribution over the K QA-pairs via softmax, and is trained to minimize negative log-likelihood of the correct QA-pair.

We obtain training data in the following manner: For a training QA-pair, we retrieve the top $2K$ QA-pairs from PAQ using RePAQ's retriever. If one of the retrieved QA-pairs has the correct answer, we treat it as a positive, and randomly sample K-1 of the incorrect retrieved questions as negatives. We train with K=10, and rerank 50 QA-pairs at test time. The reranker improves accuracy at the expense of speed. However, as QA-pairs consist of fewer tokens than passages, the reranker is still faster than retrieve-and-read models, even for architectures such as ALBERT-xxlarge.

5 Results

We first examine the PAQ resource in general, before exploring how both CBQA models and RePAQ perform using PAQ, comparing to recently published systems. We measure performance using Natural Questions (NQ, Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017), evaluating using standard Exact Match (EM) score.

[3] Other methods, such as heuristically constructing paraphrase pairs assuming that questions with the same answer are paraphrases, and training with sampled negatives would also be valid, but were not competitive in early experiments

[4] We could use $p_{\texttt{gen}}$ as a reranker/aggregator for QA, but in practice find it both slower and less accurate than the reranker described in Section 4.2.2

5.1 Examining PAQ

We generate PAQ by applying the pipeline described in Section 3 to the Wikipedia dump from Karpukhin et al. (2020), which splits Wikipedia into 100-word passages. We use the passage selection model p_s to rank all passages, generate from the top 10M, before applying global filtering.[5]

We are interested in understanding the effectiveness of different answer extractors, and whether generating more questions per answer span leads to better results. To address these questions, we create three versions of PAQ, described below. PAQ_L uses a learned answer extractor, and a question generator trained on NQ and TriviaQA. We extract 8 answers per passage and use beam size 4 for question generation. In $\text{PAQ}_{L,1}$ we only use the top scoring question in the beam, whereas in $\text{PAQ}_{L,4}$ we use all four questions from the beam, allowing several questions to be generated from each answer span. $\text{PAQ}_{NE,1}$ uses the NER answer extractor, and a generator trained on NQ. $\text{PAQ}_{NE,1}$ allow us to assess whether diversity in the form of answer extractors and question generators gives better results. The final KB, referred to as just ''PAQ'', is the union of PAQ_L and PAQ_{NE}.

As shown in Table 1, PAQ consists of 65M filtered QA pairs.[6] This was obtained by extracting 165M answer spans and generating 279M unique questions before applying global filtering. Table 1 shows that the PAQ_L pipeline is more efficient than PAQ_{NE}, with 24.4% of QA-pairs surviving filtering, compared to 18.6%.

PAQ Answer Coverage To evaluate answer extractors, we calculate how many answers in the validation sets of TriviaQA and NQ also occur in PAQ's filtered QA-pairs. Table 1 shows that the answer coverage of PAQ is very high—over 90% for both TriviaQA and NQ. Comparing PAQ_L with PAQ_{NE} shows that the learnt extractor achieves higher coverage, but the union of the two leads to the highest coverage overall. Comparing $\text{PAQ}_{L,1}$ and $\text{PAQ}_{L,4}$ indicates that using more questions from the beam also results in higher coverage.

[5]Generation was stopped when downstream performance with RePAQ did not significantly improve.

[6]Each question only has one answer due to global filtering.

Dataset	Extracted Answers	Unique Qs	Filtered QAs	Ratio	Coverage NQ	Coverage TQA
$\text{PAQ}_{L,1}$	76.4M	58.0M	14.1M	24.4%	88.3	90.2
$\text{PAQ}_{L,4}$	76.4M	225.2M	53.8M	23.9%	89.8	90.9
$\text{PAQ}_{NE,1}$	122.2M	65.4M	12.0M	18.6%	83.5	88.3
PAQ	165.7M	279.2M	64.9M	23%	90.2	91.1

Table 1: PAQ dataset statistics and ODQA dataset answer coverage. ''Ratio'' refers to the number of generated questions which pass the global consistency filter.

PAQ Question Generation Quality Illustrative examples from PAQ can be seen in Table 2. Manual inspection of 50 questions from PAQ reveals that 82% of questions accurately capture information from the passage and contain sufficient details to locate the answer. Sixteen percent of questions confuse the semantics of certain answer types, either by conflating similar entities in the passage or by misinterpreting rare phrases (see examples 7 and 8 in Table 2). Finally, we find small numbers of grammar errors (such as example 5) and mismatched wh-words (5% and 2%, respectively).[7]

Other Observations PAQ often contains several paraphrases of the same QA-pair. This redundancy reflects how information is distributed in Wikipedia, with facts often mentioned on several different pages. Generating several questions per answer span also increases redundancy. Although this means that PAQ could be more information-dense if a de-duplication step was applied, we later show that RePAQ always improves with more questions (Section 5.2.1). This suggests that it is worth increasing redundancy for greater coverage.

5.2 Question Answering Results

In this section, we shall compare how the PAQ-leveraging models proposed in Section 4 compare to existing approaches. We primarily compare to a state-of-the-art retrieve-and-read model, Fusion-in-Decoder (FiD, Izacard and Grave, 2021). FiD uses DPR (Karpukhin et al., 2020) to retrieve passages from Wikipedia, and feeds them into T5 (Raffel et al., 2020) to generate a final answer.

Table 3 shows the highest-accuracy configurations of our models alongside recent state-of-the-art models. We make the following

[7]Further details in Appendix A.3.

#	Question	Answer	Comment
1	who created the dutch comic strip panda	Martin Toonder	✓
2	what was the jazz group formed by john hammond in 1935	Goodman Trio	✓
3	astrakhan is russia's main market for what commodity	fish	✓
4	what material were aramaic documents rendered on	leather	✓
5	when did the giant panda chi chi died	22 July 1972	✓, Grammar error
6	pinewood is a village in which country	England	∼, Also a Pinewood village in USA
7	who was the mughal emperor at the battle of lahore	Ahmad Shah Bahadur	✗ Confuses with Ahmad Shah Abdali
8	how many jersey does mitch richmond have in the nba	2	✗ His Jersey No. was 2

Table 2: Representative examples from PAQ. ✓indicates correct, ∼ ambiguous, and ✗ incorrect facts, respectively.

#	Model Type	Model	NaturalQuestions	TriviaQA
1	Closed-book	T5-11B-SSM (Roberts et al., 2020)	35.2	51.8
2	Closed-book	BART-large (Lewis et al., 2021)	26.5	26.7
3	QA-pair retriever	Dense retriever (Lewis et al., 2021)	26.7	28.9
4	Open-book, retrieve-and-read	RAG-Sequence (Lewis et al., 2020b)	44.5	56.8
5	Open-book, retrieve-and-read	FiD-large, 100 docs (Izacard and Grave, 2021)	51.4	**67.6**
6	Open-book, phrase index	DensePhrases (Lee et al., 2021)	40.9	50.7
7	Closed-book	BART-large, pre-finetuned on PAQ	32.7	33.2
8	QA-pair retriever	RePAQ (retriever only)	41.2	38.8
9	QA-pair retriever	RePAQ (with reranker)	<u>47.7</u>	50.7
10	QA-pair retriever	RePAQ-multitask (retriever only)	41.7	41.3
11	QA-pair retriever	RePAQ-multitask (with reranker)	47.6	<u>52.1</u>
12	QA-pair retriever	RePAQ-multitask w/ FiD-Large Backoff	**52.3**	67.3

Table 3: Exact Match score for highest accuracy RePAQ configurations in comparison to recent state-of-the-art systems. Highest score indicated in bold, highest non-retrieve-and-read model underlined.

observations: Comparing rows 2 and 7 shows that a CBQA BART model trained with PAQ outperforms a comparable NQ-only model by 5%, and is only 3% behind T5-11B (row 1) which has 27x more parameters. Second, we note strong results for RePAQ on NQ (row 9), outperforming retrieve-and-read systems such as RAG by 3% (row 4).

Multitask training RePAQ on NQ and TriviaQA improves TriviaQA results by 1%-2% (comparing rows 8-9 with 10-11). RePAQ does not perform as strongly on TriviaQA (see Section 5.2.6), but is within 5% of RAG, and outperforms concurrent work on real-time QA, DensePhrases (row 6, Lee et al., 2021). Lastly, row 12 shows that combining RePAQ and FiD-large into a combined system is 0.9% more accurate than FiD-large (see Section 5.2.4 for more details).

5.2.1 Ablating PAQ Using RePAQ

Table 4 shows RePAQ's accuracy using different PAQ variants. To establish the effect of filtering,

#	KB	Filtering	Size	Exact Match Retrieve	Exact Match Rerank
1	NQ-Train	–	87.9K	27.9	31.8
2	$PAQ_{L,1}$	None	58.0M	21.6	30.6
3	$PAQ_{L,1}$	Local	31.7M	28.3	34.9
4	$PAQ_{L,1}$	Global	14.1M	38.6	44.3
5	$PAQ_{L,4}$	Global	53.8M	40.3	45.2
6	$PAQ_{NE,1}$	Global	12.0M	37.3	42.6
7	PAQ	Global	64.9M	**41.6**	**46.4**

Table 4: The effect of different PAQ subsets on the NQ validation accuracy of RePAQ.

we evaluate RePAQ with unfiltered, locally filtered and globally filtered QA-pairs on $PAQ_{L,1}$. Rows 2-4 show that global filtering is crucial, leading to a 9% and 14% increase over locally filtered and unfiltered QA-pairs, respectively.

We also note a general trend in Table 4 that adding more globally filtered questions improves accuracy. Rows 4-5 show that using four

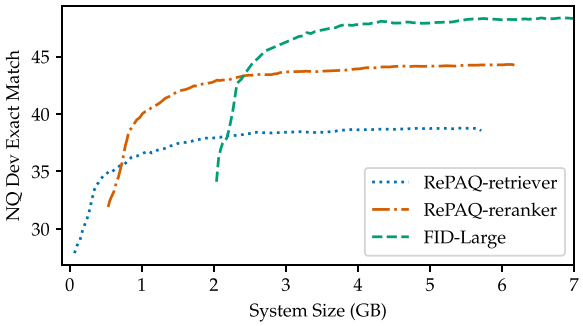

Figure 2: Size vs. accuracy for RePAQ and FiD-large as a function of the number of items in the index.

Model	Retriever	Reranker	Exact Match	Q/sec
FiD-large	–	–	51.4	0.5
FiD-base	–	–	48.2	2
RePAQ	base	–	40.9	1400
RePAQ	xlarge	–	41.5	800
RePAQ	base	base	45.7	55
RePAQ	xlarge	xxlarge	47.6	6

Table 5: Inference speeds of various configurations of RePAQ compared to FiD models on NQ.

questions per answer span is better than generating one (+0.9%), and rows 5-7 show that combining PAQ_{NE} and PAQ_{L} also improves accuracy (+1.2%). Empirically we did not observe any cases where increasing the number of globally filtered QA-pairs reduced accuracy, even when there were millions of QA-pairs already.

5.2.2 System Size vs. Accuracy

PAQ's QA-pairs are accompanied by scores of how likely they are to be asked. These scores can be used to filter the KB and reduce the RePAQ system size. A similar procedure can be used to filter the background corpus for a retrieve-and-read model (Izacard et al., 2020). We compare the system size of a FiD-large system and RePAQ as the number of items (passages and QA-pairs, respectively) in their indexes are reduced. We select which passages and QA-pairs are included using the passage selection model p_s.[8] Further experimental details can be found in Appendix A.4. Figure 2 shows that both system sizes can be reduced several-fold with only a small drop in accuracy, demonstrating the effectiveness of p_s. FiD can achieve a higher accuracy, but requires larger system sizes. RePAQ can be reduced to a smaller size before a significant accuracy drop, driven primarily by the higher information density of QA-pairs relative to passages, and fewer model parameters used by RePAQ compared to FiD.

5.2.3 Inference Speed vs. Accuracy

We train a variety of differently sized RePAQ models to explore the relationship between accuracy and inference speed. We use a fast Hierarchical Navigable Small World (HNSW) index in FAISS (Malkov and Yashunin, 2020; Johnson et al., 2019)[9] and measure the time required to evaluate the NQ test set on a system with access to one GPU.[10] Table 5 shows these results. Some retriever-only RePAQ models can answer over 1,000 questions per second, and are relatively insensitive to model size, with ALBERT-base only scoring 0.5% lower than ALBERT-xlarge. They also outperform retrieve-and-read models like REALM (40.4%, Guu et al., 2020) and recent real-time QA models like DensePhrases (40.9%, Lee et al., 2021). We find that larger, slower RePAQ rerankers achieve higher accuracy. However, even the slowest RePAQ is 3x faster than FiD-base, while only being 0.8% less accurate, and 12x faster than FiD-large.

5.2.4 Selective Question Answering

Models should not just be able to answer accurately, but also ''know when they don't know'', and abstain when they are unlikely to produce good answers (Voorhees, 2002). This task is challenging for current systems (Asai and Choi, 2020; Jiang et al., 2020b), and has been approached in MRC by training on unanswerable questions (Rajpurkar et al., 2018) and for trivia systems by using incremental QA (Rodriguez et al., 2019).

We find that RePAQ's retrieval and reranking scores are well-correlated with answering correctly. RePAQ can thus be used for selective QA by abstaining when the score is below a certain threshold. Figure 3 shows a *risk-coverage* plot (Wang et al., 2018) for RePAQ and FiD, where

[8]Here, we use PAQ_{L1}, which is 5x smaller than the full PAQ, but retains most of the accuracy (see Table 4).

[9]The HNSW index has negligible (∼0.1%) drop in retriever accuracy compared to a flat index.

[10]System details can be found in Appendix A.5.

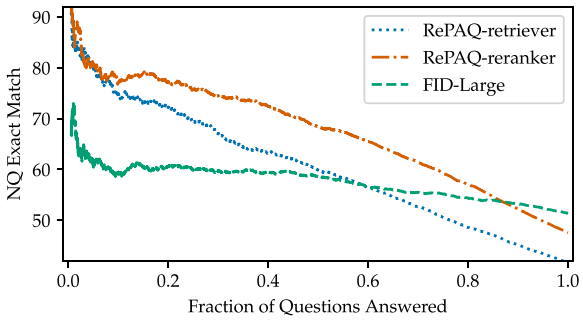

Figure 3: Risk-coverage plot for FiD and RePAQ.

we use FiD's answer log probability for its answer confidence.[11] The plot shows the accuracy on the top N% highest confidence answers for NQ. If we require models to answer 75% of user questions, RePAQ's accuracy on the questions it does answer is 59%, whereas FiD, which has poorer calibration, scores 55%. This difference is even more pronounced with stricter thresholds—with coverage of 50%, RePAQ outperforms FiD by over 10%. FiD only outperforms RePAQ when we require systems to answer over 85% of questions.

Although RePAQ's selective QA is useful in its own right, it also allows us to combine the slow but accurate FiD with the fast and precise RePAQ, which we refer to as *backoff*. We first try to answer with RePAQ, and if the confidence is below a threshold determined on validation data, we pass the question onto FiD. For NQ, the combined system is 2.1x faster than FiD-large, with RePAQ answering 57% of the questions, and the overall accuracy is 1% higher than FiD-large (see Table 3).

If inference speed is a priority, the threshold can be decreased so that RePAQ answers 80% of the questions, which retains the same overall accuracy as FiD, with a 4.6x speedup. For TriviaQA, the combined system backs off to FiD earlier, due to the stronger relative performance of FiD. Additional details can be found in Appendix A.6.

5.2.5 Analyzing RePAQ's Predictions

Some examples of top retrieved questions are shown in Table 6. When RePAQ answers correctly, the retrieved question is a paraphrase of the test question from PAQ in 89% of cases. As such,

[11] We also investigate improving FiD's calibration using an auxiliary model, see Appendix A.6. We find that the most effective way to calibrate FiD is to use RePAQ's confidences.

Input: who was the film chariots of fire about	**A**: Eric Liddell	
who was the main character in chariots of fire	**A**: *Eric Liddell*	✓
who starred in the movie chariots of fire	**A**: Ian Charleson	✗
which part did straan rodger play in chariots of fire	**A**: Sandy McGrath	✗
who played harold in the 1981 film chariots of fire	**A**: Ben Cross	✗
who is the main character in chariots of fire	**A**: Eric Liddell	✓
Input: what is the meaning of the name didymus	**A**: twin	
what language does the name didymus come from	**A**: Greek	✗
where does the name didymus come from in english	**A**: Greek	✗
what does the word domus mean in english	**A**: home	✗
how long has the term domus been used	**A**: 1000s of years	✗
what does the greek word didyma mean	**A**: *twin*	✓
Input: what is the name of a group of llamas	**A**: herd	
what are llamas and alpacas considered to be	**A**: domesticated	✗
what are the figures of llamas in azapa valley	**A**: Atoca	✗
what are the names of the llamas in azapa valley	**A**: Atoca	✗
what is the scientific name for camels and llamas	**A**: *Camelidae*	✗
are llamas bigger or smaller than current forms	**A**: larger	✗

Table 6: Examples of top 5 retrieved QA-pairs for NQ. Italics indicate QA-pairs chosen by reranker.

there is high (80.8 ROUGE-L) similarity between correctly answered test questions and the top retrieved questions. Nine percent of test questions even exist verbatim in PAQ, and are thus trivial to answer. The reranker primarily improves over the retriever for ambiguous cases, and cases where the top retrieved answer does not have the right granularity. In 32% of cases, RePAQ does not retrieve the correct answer in the top 50 QA-pairs, suggesting a lack of coverage may be a significant source of error. In these cases, retrieved questions are much less similar to the test question than for correctly answered questions, dropping by 20 ROUGE-L. We also observe cases where retrieved questions match the test question, but the answer does not match the desired answer. This is usually due to different answer granularity, but in a small number of cases was due to factually incorrect answers.

5.2.6 Does the Filtering Model Limit RePAQ's Accuracy?

As RePAQ relies on retrieving paraphrases of test questions, we may expect that the ODQA filtering model places an upper bound on its performance. For example, if a QA-pair is generated that overlaps with a test QA-pair, but the filter cannot answer it correctly, that QA-pair will not be added to PAQ, and RePAQ cannot use it to answer the test question. The NQ FiD-base-50-doc model used for filtering scores 46.1% and 53.1% for NQ and TriviaQA, respectively. RePAQ actually outperforms the filter model on NQ by 1.6%. This is possible because generated questions can be phrased in such a way that they are easier to answer, for example, being less ambiguous

Model	Total	Q Overlap	A-only Overlap	No Overlap
CBQA BART w/ NQ	26.5	67.6	10.2	0.8
CBQA BART w/ NQ+PAQ	28.2	52.8	24.4	9.4
+ final NQ finetune	32.7	69.8	22.2	7.51
RePAQ (retriever only)	41.7	65.4	31.7	21.4
RePAQ (with reranker)	47.3	73.5	39.7	26.0

Table 7: NQ Behavioural splits (Lewis et al., 2021). ''Q overlap'' are test questions with paraphrases in training data. ''A-only'' are test questions where answers appear in training data, but questions do not. ''No overlap'' where neither question or answer overlap.

(Min et al., 2020b). RePAQ *can* then retrieve the paraphrased QA-pair and answer correctly, even if the filter could not answer the test question directly. The filtering model's weaker scores on TriviaQA helps explain why RePAQ is not as strong on this dataset. We speculate that a stronger filtering model for TriviaQA would in turn improve RePAQ's results.

5.3 Closed-book QA vs. RePAQ

Table 7 shows results on test set splits that measure how effectively models memorize QA-pairs from the NQ train set (''Q overlap''), and generalize to novel questions (''A overlap only'' and ''No overlap'').[12] Comparing CBQA models trained on NQ vs. those trained on NQ and PAQ show that models trained with PAQ answer more questions correctly from the ''A-only overlap'' and ''No overlap'' categories, indicating they learned facts not present in the NQ train set. Applying further NQ finetuning on the PAQ CBQA model improves scores on ''Q overlap'' (indicating greater memorisation of NQ), but scores on the other categories drop (indicating reduced memorization of PAQ). However, RePAQ, which explicitly retrieves from PAQ rather than memorizing it in parameters, strongly outperforms the CBQA model in all categories, demonstrating that the CBQA model struggles to memorize enough facts from PAQ. Larger CBQA models should be better able to memorise PAQ, but have downsides in terms of system resources. Future work should address how to better store PAQ in CBQA model parameters.

[12]See Lewis et al. (2021) for further details.

6 Related Work

ODQA has a been a topic of interest for atleast five decades (Simmons, 1965), with its modern formulation established by TREC in the early 2000s (Voorhees, 1999). For a detailed history, the reader is referred to Chen and Yih (2020). Interest in ODQA has recently intensified for its practical applications and for measuring how well models store and access knowledge (Petroni et al., 2021).

KBQA A number of early approaches in ODQA focused on using structured KBs (Berant et al., 2013) such as Freebase (Bollacker et al., 2008), with recent examples from Févry et al. (2020) and Verga et al. (2020). This approach often has high precision but suffers when the KB does not match user requirements, or where the schema limits what knowledge can be stored. We populate our KB with semi-structured QA-pairs that are specifically designed to be relevant at test time, mitigating these drawbacks, while sharing benefits such as precision and extensibility.

OpenIE Our work touches on KB construction and open information extraction (OpenIE) (Angeli et al., 2015). Here, the goal is to extract structured or semi-structured facts from text, typically (subject, relation, object) triples for use in tasks such as slot-filling (Surdeanu, 2013). We generate natural language QA-pairs rather than OpenIE triples, and do not attempt to extract all possible facts in a corpus, focusing only on those likely to be asked. QA-pairs have also been used in semantic role labeling, for example, QA-SRL (FitzGerald et al., 2018).

Real-time ODQA Systems prioritizing fast runtime over accuracy are sometimes referred to as *real-time QA* systems (Seo et al., 2018). DenSPI (Seo et al., 2019) and a contemporary work, DensePhrases (Lee et al., 2021), index all possible phrases in a corpus, and learn mappings from questions to passage-phrase pairs. We also build an index for fast answering, but generate and index globally answerable questions. Indexing QA-pairs can be considered as indexing summaries of important facts from the corpus, rather than indexing the corpus itself. We also generate and store multiple questions per passage-answer pair, relieving information bottlenecks from encoding a passage-answer pair into a single vector.

Question Generation for QA Question generation has been used for various purposes, such as data augmentation (Alberti et al., 2019; Lewis et al., 2019; Lee et al., 2021), improved retrieval (Nogueira et al., 2019), generative modelling for contextual QA (Lewis and Fan, 2018), as well as being studied in its own right (Du et al., 2017; Hosking and Riedel, 2019). Serban et al. (2016) generate large numbers of questions from Freebase, but do not address how to use them for QA. Closest to our work is the recently proposed OceanQA (Fang et al., 2020). OceanQA first generates contextual QA-pairs from Wikipedia. At test-time, a document retrieval system is used to retrieve the most relevant passage for a question and the closest pre-generated QA-pair from that passage is selected. In contrast, we focus on generating a large KB of non-contextual, globally consistent ODQA questions and explore what QA systems are facilitated by such a resource.

7 Discussion and Conclusion

We have introduced a dataset of 65M QA-pairs, and explored its uses for improving ODQA models. We demonstrated the effectiveness of RePAQ, a system that retrieves from PAQ, in terms of accuracy, speed, space efficiency and selective QA.

We found that RePAQ's errors are driven by a lack of coverage, thus generating more QA-pairs should improve accuracy further. However, phenomena such as compositionality may impose practical limits on this approach. Multi-hop RePAQ extensions suggest themselves as ways forward here, as well as back-off systems (see Section 5.2.4). Generating PAQ is also computationally intensive due to its large scale and global filtering, but it should be a useful, re-usable resource for researchers. Nevertheless, future work should be carried out to improve the efficiency of generation to expand PAQ's coverage.

We also demonstrated PAQ's utility for improved CBQA, but note a large accuracy gap between our CBQA models and RePAQ. Exploring the trade-offs between storing and retrieving knowledge parametrically or non-parametrically is of great current interest (Lewis et al., 2020b; Cao et al., 2021), and PAQ should be a useful testbed for probing this relationship further. We also note that PAQ could be used as general data-augmentation when training any open-domain QA model or retriever. We consider such work out-of-scope here, but leveraging PAQ to improve other models should be explored in future work.

Acknowledgments

The authors would like to extend their gratitude to the anonymous reviewers and Action Editor for their highly detailed and insightful comments and feedback. The authors would also like to thank Gautier Izacard, Ethan Perez, Max Bartolo, Tom Kwiatkowski, and Jimmy Lin for helpful discussions and feedback on the project. PM and PS are supported by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 875160.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Akari Asai and Eunsol Choi. 2020. Challenges in information seeking QA: Unanswerable questions and paragraph retrieval. *arXiv:2010.11915 [cs]*. ArXiv: 2010.11915.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1247–1250, Vancouver, Canada. Association for Computing Machinery.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

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, Vancouver, Canada. Association for Computational Linguistics.

Danqi Chen and Wen-tau Yih. 2020. Open-domain question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 34–37, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conferenceof 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.

Pedro Domingos. 2020. Every Model Learned by Gradient Descent Is Approximately a Kernel Machine. *arXiv:2012.00152 [cs, stat]*. ArXiv: 2012.00152.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Yuwei Fang, Shuohang Wang, Zhe Gan, Siqi Sun, and Jingjing Liu. 2020. Accelerating real-time question answering via question generation. *arXiv:2009.05167 [cs]*. ArXiv: 2009.05167.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale QA-SRL parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2051–2060, Melbourne, Australia. Association for Computational Linguistics.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4937–4951, Online. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938, PMLR.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Tom Hosking and Sebastian Riedel. 2019. Evaluating rewards for question generation models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2278–2283, Minneapolis, Minnesota. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, USA. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

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: 2012.15156 [cs]*. ArXiv: 2012.15156.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. How can we know when language models know? *arXiv:2012 .00955 [cs]*. ArXiv: 2012.00955.

Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2020b. Generalizing natural language analysis through span-relation representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2120–2133, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, pages 1–1.

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.

H. Jégou, M. Douze, and C. Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128. https://doi.org/10.1109/TPAMI.2010.57

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 7:452–466. https://doi.org/10.1162/tacl_a_00276

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. *arXiv:2012.12624 [cs]*. ArXiv: 2012.12624.

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.

Mike Lewis and Angela Fan. 2018. Generative question answering: Learning to answer the whole question. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online. Association for Computational Linguistics.

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: 1907.11692 [cs]*. ArXiv: 1907.11692.

Yu A. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen-tau Yih. 2020a. NeurIPS 2020 EfficientQA Competition: Systems, analyses and lessons learned. *arXiv:2101.00133 [cs]*. ArXiv: 2101.00133.

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

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv:1904.08375 [cs]*. ArXiv: 1904.08375.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: A benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Pedro Rodriguez, Shi Feng, Mohit Iyyer, He He, and Jordan Boyd-Graber. 2019. Quizbowl: The case for incremental question answering. *arXiv:1904.04792 [cs]*. ArXiv: 1904.04792.

Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 559–564, Brussels, Belgium. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany. Association for Computational Linguistics.

R. F. Simmons. 1965. Answering English questions by computer: A survey. *Communications of the Association for Computing Machinery*, 8(1):53–70. https://doi.org/10.1145/363707.363732

M. Surdeanu. 2013. Overview of the TAC2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling. *TAC*.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William W. Cohen. 2020. Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *arXiv:2007.00849 [cs]*. ArXiv: 2007.00849.

Ellen M. Voorhees. 1999. The TREC-8 Question Answering Track Report. In *Proceedings* of TREC-8, pages 77–82.

Ellen M. Voorhees. 2002. Overview of the TREC 2002 question answering track. In *Proceedings of The Eleventh Text REtrieval Conference, TREC* 2002, Gaithersburg, Maryland, USA, November 19–22, 2002, volume 500–251 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Ellen M. Voorhees and Donna K. Harman, editors. 1999. *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17–19, 1999*, volume 500–246 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

William Wang, Angelina Wang, Aviv Tamar, Xi Chen, and Pieter Abbeel. 2018. Safer classification by synthesis. *arXiv:1711.08534 [cs, stat]*. ArXiv: 1711.08534.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.

Jinfeng Xiao, Lidan Wang, Franck Dernoncourt, Trung Bui, Tong Sun, and Jiawei Han. 2021. Open-domain question answering with pre-constructed question spaces. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 61–67, Online. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. Data augmentation for BERT fine-tuning in open-domain question answering. *arXiv:1904.06652 [cs]*. ArXiv: 1904.06652.

Qinyuan Ye, Belinda Z. Li, Sinong Wang, Benjamin Bolte, Hao Ma, Wen-tau Yih, Xiang Ren, and Madian Khabsa. 2021. Studying strategically: Learning to mask for closed-book QA. *arXiv:2012.15856 [cs]*. ArXiv: 2012.15856.

A Appendices

A.1 Details on Dataset Splits

For NQ we use the standard open-domain splits introduced by Lee et al. (2019), and train-development splits used by Karpukhin et al. (2020). For TriviaQA, we use the standard open-domain splits, which correspond to the unfiltered-train and unfiltered-dev reading comprehension splits (Joshi et al., 2017; Lee et al., 2019).

A.2 Further Details on Passage Selection

The passage selection model is based on $\text{RoBERTa}_{\text{BASE}}$ (Liu et al., 2019). We feed each passage into the model and use an MLP on top of the [CLS] representation to produce a score. We use this model to obtain a score for every passage in the corpus. The top N highest-scoring passages are selected for QA-pair generation. This model achieves 84.7% recall on the NQ dev set.

A.3 Further Details on Question Quality

For NQ, we find that the retrieved questions are paraphrases of the test questions in the majority of cases. We conduct human evaluation on 50 random sampled questions generated from the wikipedia passage pool. We make the following observations: i) 82% of questions accurately capture the context of the answer in the passage, and contain sufficient details to locate the answer. ii) 16% of questions have incorrect semantics with respect to their answers. These errors are driven by two main factors: *Mistaking extremely similar entities* and *Generalization to rare phrases*. An example of the former is ‘‘what is the eastern end of the Kerch peninsula’’ for the passage ‘‘The Kerch Peninsula is located at the eastern end of the Crimean Peninsula’’ and the answer ‘‘the Crimean Peninsula’’. An example of the latter is where the model interprets digits separated by colons as date ranges, such as for the passage ‘‘under a 109–124 loss to the Milwaukee Bucks’’, the question is generated as ‘‘when did . . . play for the Toronto Raptors’’. iii) Only 2% of questions mismatch question wh-words in the analysis sample.

A.4 Further Details on System Size vs. Accuracy

The experiment in Section 5.2.2 measures the bytes required to store the models, the text of the documents/QA-pairs, and a dense index. For Figure 2, We assume models are stored at FP16 precision, the text has been compressed using LZMA,[13] and the indexes use 768 dimensional vectors, and Product Quantization (Jégou et al., 2011). These are relatively standard settings when building efficient systems (Izacard et al., 2020; Min et al., 2020a). The RePAQ model used here consists of an ALBERT-base retriever and

[13] `https://tukaani.org/xz/`.

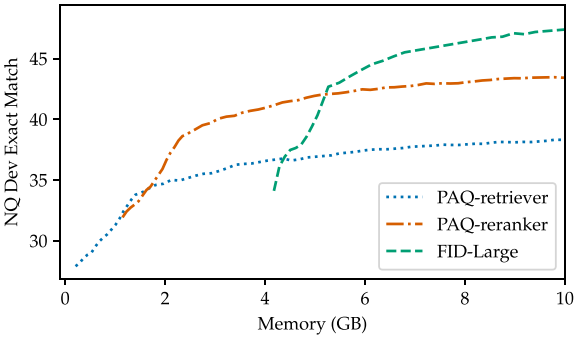

Figure 4: System size vs. accuracy for RePAQ and FiD as a function of the number of items in the index, with different experimental setup than the main paper.

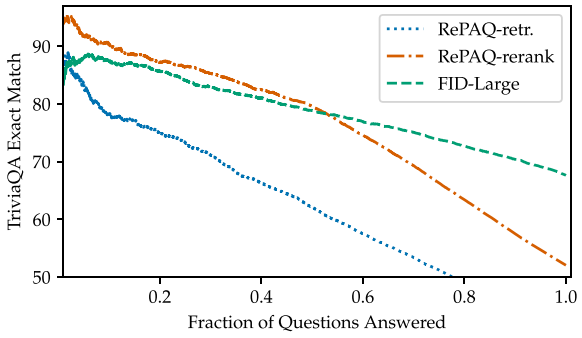

Figure 5: Risk-coverage plot on TriviaQA. FiD has higher overall accuracy, but RePAQ's reranker still performs best for coverages <50%.

ALBERT-xxlarge reranker, and the FiD system consists of DPR (Karpukhin et al., 2020) (consisting of two BERT-base retrievers) and a T5-large reader (Raffel et al., 2020). Using a different setup (e.g., storing models at full precision, no text compression, and FP16 index quantization, shown in Figure 4) shifts the relative position of the curves in Figure 2, but the qualitative relationship is unchanged.

A.5 Further Details on Inference Speed

The machine used for speed benchmarking is a machine learning workstation with 80 CPU cores, 512GB of CPU RAM and access to one 32GB NVIDIA V100 GPU. Inference is carried out at mixed precision for all systems, and questions are allowed to be answered in parallel. Models are implemented in PyTorch (Paszke et al., 2019) using Transformers (Wolf et al., 2020). Measurements are repeated 3 times and the mean time is reported, rounded to an appropriate significant figure. The HNSW index used in this experiment indexes all 65M PAQ QA-pairs with 768 dimensional vectors, uses an `ef_construction` of 80, `ef_search` of 32, and `store_n` of 256, and performs up to 2048 searches in parallel. This index occupies 220GB, but can be considerably compressed with scalar or product quantization, or training retrievers with smaller dimensions – see Section A.8 for details of such an index.

A.6 Further Details on Selective QA

Figure 5 shows the risk-coverage plot for TriviaQA. The results are qualitatively similar to NQ (Figure 3), although FiD's stronger overall performance shifts its risk-coverage curve up the accuracy axis relative to RePAQ. FiD also appears better calibrated on TriviaQA than it is for NQ, indicated by higher gradient. However, RePAQ remains better calibrated, outperforming it for answer coverages below 50%.

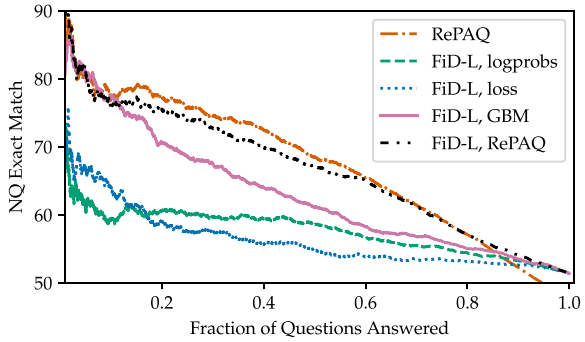

Figure 6: Risk-coverage plot for different calibration methods for FiD (RePAQ included for comparison). Using RePAQ's confidence scores to calibrate FiD leads to FiD's strongest results.

We also investigate improving FiD's calibration on NQ, using a post-hoc calibration technique similar to Jiang et al. (2020a). We train a Gradient Boosting Machine (GBM; Friedman, 2001) on development data to predict whether FiD has answered correctly or not. The GBM is featurized with FiD's answer loss, answer log probability and the retrieval score of the top 100 retrieved documents from DPR. Figure 6 shows these results. We first note that FiD-Large's answer loss and answer log probabilities perform similarly, and both struggle to calibrate FiD, as mentioned in the main paper. The GBM improves calibration, especially at lower coverages, but still lags behind RePAQ by 7% EM at 50% coverage. We

also note that we can actually use RePAQ's confidence scores to calibrate FiD. Here, we use FiD's predicted answer, but RePAQ's confidence score to decide whether to answer or not. This result is also plotted in Figure 6, and results in FiD's best risk-coverage curve. Despite these improvements, FiD is still not as well-calibrated as RePAQ.

A.7 Additional Model Training Details

RePAQ models were trained for up to 3 days on a machine with 8 NVIDIA 32GB V100 GPUs. Validation Exact Match score was used to determine when to stop training in all cases. RePAQ retrievers were trained using Fairseq (Ott et al., 2019), and rerankers were trained in Transformers (Wolf et al., 2020) in PyTorch (Paszke et al., 2019). The PAQ CBQA models were trained in Fairseq for up to 6 days on 8 NVIDIA 32GB V100 GPUs, after which validation accuracy had plateaued. Hyperparameters were tuned to try to promote faster learning, but learning became unstable with learning rates greater than 0.0001.

A.8 Memory-Efficient REPAQ Retriever

Code, models, and data are available at `https://github.com/facebookresearch/PAQ`. As part of this release, we have trained a memory-efficient RePAQ retriever designed for use with more modest hardware than the main RePAQ models. This consists of an ALBERT-base retriever, with 256-dimensional embedding, rather than the 768-dimensional models in the main paper. We provide 2 FAISS indices (Johnson et al., 2019) for use with this model, both built with 8-bit scalar quantization. The first index is a flat index, which is very memory-friendly, requiring only 16GB of CPU RAM, but relatively slow (1-10 questions per second). The other is an HNSW approximate index (Malkov and Yashunin, 2020), requiring $\sim$32 GB of CPU RAM, but can process 100-1000 questions per second. This memory-efficient system is highly competitive with the models in the main paper, actually outperforming the AlBERT-base model (+0.6%, NQ, +0.5%, TQA), and only trailing the AlBERT-xlarge model by 0.6% on average (-0.3% NQ, -0.9% TQA).