ReQA: An Evaluation for End-to-End Answer Retrieval Models

Amin Ahmad, Noah Constant, Yinfei Yang, Daniel Cer
Google Research, Mountain View, USA
{aahmad, nconstant, yinfeiy, cer}@google.com

Abstract

Popular QA benchmarks like SQuAD have driven progress on the task of identifying answer spans within a specific passage, with models now surpassing human performance. However, retrieving relevant answers from a huge corpus of documents is still a challenging problem, and places different requirements on the model architecture. There is growing interest in developing scalable answer retrieval models trained end-to-end, bypassing the typical document retrieval step. In this paper, we introduce Retrieval Question-Answering (ReQA), a benchmark for evaluating large-scale sentence-level answer retrieval models. We establish baselines using both neural encoding models as well as classical information retrieval techniques. We release our evaluation code to encourage further work on this challenging task.

1 Introduction

Popular QA benchmarks like SQuAD (Rajpurkar et al., 2016) have driven impressive progress on the task of identifying spans of text within a specific passage that answer a posed question. Recent models using BERT pretraining (Devlin et al., 2019) have already surpassed human performance on SQuAD 1.1 and 2.0.

While impressive, these systems are not yet sufficient for the end task of answering user questions at scale, since in general, we don't know which documents are likely to contain an answer. On the one hand, typical document retrieval solutions fall short here, since they aren't trained to directly model the connection between questions and answers in context. For example, in Figure 1, a relevant answer appears on the Wikipedia page for New York, but this document is unlikely to be retrieved, as the larger document is not highly relevant to the question. On the other hand,

Question: Which US county has the densest population?

Wikipedia Page: New York City

Answer: Geographically co-extensive with New York County, the borough of Manhattan's 2017 population density of 72,918 inhabitants per square mile (28,154/km^2) makes it the highest of any county in the United States and higher than the density of any individual American city.

Figure 1: A hypothetical example of end-to-end answer retrieval, where the document containing the answer is not "on topic" for the question.

QA models with strong performance on reading comprehension can't be used directly for large-scale retrieval. This is because competitive QA models use interactions between the question and candidate answer in the early stage of modeling (e.g. through cross-attention) making it infeasible to score a large set of candidates at inference time.

There is growing interest in training end-to-end retrieval systems that can efficiently surface relevant results without an intermediate document retrieval phase (Gillick et al., 2018; Cakaloglu et al., 2018; Seo et al., 2019; Henderson et al., 2019). We are excited by this direction, and hope to promote further research by offering the Retrieval Question-Answering (ReQA) benchmark, which tests a model's ability to retrieve relevant answers efficiently from a large set of documents. Our code is available at https://github.com/google/retrieval-qa-eval.

The remainder of the paper is organized as follows. In Section 2, we define our goals in developing large-scale answer retrieval models. Section 3 describes our method for transforming within-document reading comprehension tasks into Retrieval Question-Answering (ReQA) tasks, and details our evaluation procedure and metrics. Section 4 describes various neural and non-neural baseline models, and characterizes their performance on several ReQA tasks. Finally, Section 5 discusses related work.

2 Objectives

What properties would we like a large-scale answer retrieval model to have? We discuss five characteristics below that motivate the design of our evaluation.

First, we would like an **end-to-end** solution. As illustrated in Figure 1, some answers are found in surprising places. Pipelined systems that first retrieve topically relevant documents and then search for answer spans within only those documents risk missing good answers from documents that appear to have less overall relevance to the question.

Second, we need **efficient** retrieval, with the ability to scale to billions of answers. Here we impose a specific condition that guarantees scalability. We require the model to encode questions and answers *independently* as high-dimensional (e.g. 512d) vectors, such that the relevance of a QA pair can be computed by taking their dot-product, as in Henderson et al. (2017).[1] This technique enables retrieval of relevant answers using approximate nearest neighbor search, which is sub-linear in the number of documents, and in practice close to log(N). This condition rules out the powerful models like BERT that perform best on reading comprehension metrics. Note, these approaches could be used to rerank a small set of retrieved candidate answers, but the evaluation of such multi-stage systems is out of the scope of this work.

Third, we focus on **sentence-level** retrieval. In practice, sentences are a good size to present a user with a "detailed" answer, making it unnecessary to highlight specific spans for many use cases.[2] While the experiments in this paper primarily target sentence-level retrieval, we recognize that some domains may be best served by retrieval at a different granularity, such as phrase or passage. The evaluation techniques described in Section 3 can be easily extended to cover these different granularities.

Fourth, a retrieval model should be **context aware**, in the sense that the context surrounding a sentence should affect its appropriateness as an answer. For example, an ideal QA system should be able to tell that the bolded sentence in Figure 2 is a good answer to the question, since the context makes it clear that "The official language" refers to the official language of Nigeria.

Question: What is Nigeria's official language?

Answer in Context: [...] Nigeria has one of the largest populations of youth in the world. The country is viewed as a multinational state, as it is inhabited by over 500 ethnic groups, of which the three largest are the Hausa, Igbo and Yoruba; these ethnic groups speak over 500 different languages, and are identified with wide variety of cultures. **The official language is English.** [...]

Figure 2: An example from SQuAD 1.1 where looking at the surrounding context is necessary to determine the relevance of the answer sentence.

Finally, we believe a strong model should be **general purpose**, with the ability to generalize to new domains and datasets gracefully. For this reason, *we advocate using a retrieval evaluation drawn from a specific task/domain that is never used for model training.* In the case of our tasks built on SQuAD and Natural Questions (NQ), we evaluate on retrieval over the entire training sets, with the understanding that all data from these sets is off-limits for model training. Additionally, we recommend not training on any Wikipedia data, as this is the source of the SQuAD and NQ document text. However, should this latter recommendation prove impractical, then, at the very least, the use of Wikipedia during training should be noted when reporting results on ReQA, being as specific as possible as to which subset was used and in what manner. This increases our confidence that a model that evaluates well on our retrieval metrics can be applied to a wide range of open-domain QA tasks.[3]

[1]Other distance metrics are possible. Another popular option for nearest neighbor search is cosine distance. Note, models using cosine distance can still compute relevance through a dot-product, provided the final encoding vectors are L2-normalized.

[2]In cases where highlighting the relevant span within a sentence is important, a separate highlighting module could be learned that takes a retrieved sentence as input.

[3]We strongly assert that when NLP models are used in applied systems, it is generally preferable to evaluate alternative models using data that is as distinct as reasonably possible from model training data. While this is common practice in some sub-fields of NLP such as machine translation, it is still unfortunately very common to assess other NLP

3 ReQA Evaluation

In this section, we describe our method for constructing *Retrieval Question-Answering* (ReQA) evaluation tasks from existing machine reading based QA challenges. To perform this evaluation over existing QA datasets, we first extract a large pool of candidate answers from the dataset. Models are then evaluated on their ability to correctly retrieve and rank answers to individual questions using two metrics, *mean reciprocal rank* (MRR) and *recall at N* (R@N). In Eq (1), Q is the set of questions, and $rank_i$ is the rank of the first correct answer for the ith question. In Eq (2), A_i^* is the set of correct answers for the ith question, and A_i is a scored list of answers provided by the model, from which the top N are extracted.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{1}$$

$$\text{R@N} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{|\max_N(A_i) \cap A_i^*|}{|A_i^*|} \tag{2}$$

We explore using the ReQA evaluation on both SQuAD 1.1 and Natural Questions. However, the technique is general and can be applied to other datasets as well.

3.1 ReQA SQuAD

SQuAD 1.1 is a reading comprehension challenge that consists of over 100,000 questions composed to be answerable by text from Wikipedia articles. The data is organized into paragraphs, where each paragraph has multiple associated questions. Each question can have one or more answers in its paragraph.[4]

We choose SQuAD 1.1 for our initial ReQA evaluation because it is a widely studied dataset, and covers many question types.[5] To turn SQuAD into a retrieval task, we first split each paragraph into sentences using a custom sentence-breaking tool included in our public release. For the SQuAD 1.1 train set, splitting 18,896 paragraphs produces 91,707 sentences. Next, we construct an "answer index" containing each sentence as a candidate answer. The model being evaluated computes an answer embedding for each answer (using any encoding strategy), given only the sentence and its surrounding paragraph as input. Crucially, this computation must be done independently of any specific question. The answer index construction process is described more formally in Algorithm 1.

Algorithm 1 Constructing the answer index

Input: c is a representation of a dataset in SQuAD format[6]; S is a function that accepts a string of text, s, and returns a sequence of sentences, $[s_0, s_1, \cdots, s_n]$; E_a is the embedding function, which takes answer text, a, into points in $\mathbb{R}^n$.

Output: A list of $\langle$sentence, encoding$\rangle$ tuples.

```
1: function ENCODEINDEX($c$, $S$, $E_a$)
2:     I ← new list
3:     for $x$ in $c$.data do                ▷ for every passage
4:         for $p$ in $x$.paragraphs do
5:             for $s$ in $S(p.\text{context})$ do
6:                 $s_e \leftarrow E_a(s, p.\text{context})$
7:                 append $\langle s, s_e\rangle$ to I
8:     return I
```

Similarly, we embed each question using the model's question encoder, with the restriction that only the question text be used. For the SQuAD 1.1 train set, this gives around 88,000 questions.

After all questions and answers are encoded, we compute a "relevance score" for each question-answer pair by taking the dot-product of the question and answer embeddings, as shown in Al-

models on dev and test data that is very similar to its training data (e.g., harvested from the same source using a common pipeline and a common pool of annotators). This makes it more difficult to interpret claims of models approaching "human-level" performance.

[4]Typically, multiple answers come from the same sentence. For example, the question "Where did Super Bowl 50 take place?" is associated with three answers found within the sentence "The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California." The answer spans are: [Santa Clara, California], [Levi's Stadium] and [Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.].

[5]SQuAD 2.0 adds questions that have no answer in the paragraph. While these questions are useful for testing machine reading over fixed passages, their value in a large-scale retrieval evaluation is less clear. Specifically, we can't be sure that such questions aren't answered by another sentence in the larger corpus.

[6]The SQuAD JSON format consists of a top-level list of *data* elements that represent Wikipedia articles, each containing a list of *paragraphs*. Every paragraph defines a *context*, which is its text, and a corresponding list of questions and answers. For clarity, the algorithm definition uses the same names (lines 3-6).

gorithm 2. These scores can be used to rank all (around 92,000) candidate answers for each question, and compute standard ranking metrics such as mean reciprocal rank (MRR) and recall (R@k).[7]

Algorithm 2 Scoring questions and answers

Input: $Q_{[q \times n]}$ is a matrix of question embeddings in $\mathbb{R}^n$, arranged so that the i-th row, $Q[i]$, corresponds to the embedding of q_i; $A_{[a \times n]}$ is a matrix of answer embeddings, also in $\mathbb{R}^n$, derived from the answer index, I, and arranged so that the i-th row, $A[i]$, corresponds to the embedding of a_i.

Output: $R_{[q \times a]}$ a matrix of ranking data that can be used to compute metrics such as MRR and R@k. It is arranged so that i-th row is a vector of dot-product scores for q_i, that is, $[q_i \cdot a_0, q_i \cdot a_1, \cdots, q_i \cdot a_a]$

```
1: function SCORE($Q$, $A$)
2:     $S_{[q \times a]} \leftarrow QA^T$    $\triangleright$ compute dot-products
3:     $R_{[q \times a]} \leftarrow$ new matrix
4:     for $i \leftarrow 1$ to $q$ do
5:         $R[i] \leftarrow$ rankdata[8]$(S[i])$
6:     return $R$
```

3.2 ReQA NQ

Natural Questions (NQ) consists of over 320,000 examples, where each example contains a question and an entire Wikipedia article. The questions are real questions issued by multiple users to the Google search engine, for which a Wikipedia page appeared in the top five search results. The examples are annotated by humans as to whether the returned article contains an answer to the question, and if so where. For roughly 36% of examples, the article is found to contain a "short answer": a span of text (or rarely multiple spans) that directly answers the question.

Our procedure for converting NQ into a ReQA task is similar to that described for SQuAD above. We restrict to questions with a single-span short answer, contained within an HTML <P> (paragraph) block, as opposed to answers within a list or table. When applied to the NQ training set, this filtering produces around 74,000 questions. As with SQuAD, we consider the enclosing paragraph as context (available for the model in building an answer embedding), and split the paragraph into sentences. The target answer is the sentence containing the short answer span. Each sentence in the paragraph is added to the answer index as a separate answer candidate, resulting in around 240,000 candidates overall.[9]

As with ReQA SQuAD, *we advocate excluding all of Wikipedia from model training materials*. Models satisfying this restriction give us more confidence that they can be extended to perform answer retrieval in new domains.

3.3 Dataset Statistics

The number of questions and candidate answers in the ReQA SQuAD and ReQA NQ datasets is shown in Table 1. While the number of questions is similar, ReQA SQuAD has around 2.6x fewer candidate answer sentences, making it an easier task overall. This difference is due to the fact that SQuAD itself was constructed to have many different questions answered by the same Wikipedia paragraphs.

	SQuAD	**NQ**
Questions	87,599	74,097
Candidate Sentences	91,707	239,013
Candidate Paragraphs	18,896	58,699

Table 1: The number of questions and candidates in the constructed datasets ReQA SQuAD and ReQA NQ.

Table 2 lists the average number of tokens in question and sentence-level answer text, as well as the "query coverage", which is the percentage of tokens in the question that also appear in the answer. The token coverage for ReQA SQuAD is much larger than for ReQA NQ, indicating more lexical overlap between the question and answer. This is likely due to the original SQuAD construction process whereby writers "back-wrote" questions to be answerable by the given documents.

[7]Rarely, the same question is asked in different contexts. For example, the question "How tall is Mount Olympus?" appears twice in SQuAD, with answers on the pages for both Greece and Cyprus. In this case, we consider both answers correct for the purposes of our evaluation metrics.

[8]This function assigns ranks to data, in this case assigning 1 to the largest dot-product, 2 to the second-largest dot-product, and so forth. For more details, see scipy.stats.rankdata.

[9]Since NQ includes the entire Wikipedia article, we could consider adding all sentences from *all* paragraphs as candidate answers. However even restricting to sentences from paragraphs containing short answers already produced a large index and challenged existing models, so we opted not to increase the search space further.

By comparison, NQ questions are naturally occurring anonymized, aggregated search queries, where users had no access to the answering document ahead of time.

Table 3 shows the distribution of question types for each dataset. Nearly half (47.7%) of ReQA SQuAD questions are *what* questions, with the next most frequent being *who* (9.6%) and *how* (9.3%). ReQA NQ is more balanced across question types, with the leading types being *who* (32.6%), *when* (20.3%) and *what* (15.3%).

We note that neither dataset contains many *why* questions. Performing well on this type of question may require additional reasoning ability, so it would be interesting to explore *why* questions further through more targeted ReQA datasets.

	SQuAD	NQ
Average Length (tokens)		
Question	10.1	9.1
Answer	24.0	22.9
Query Coverage (%)		
Mean	31.7	24.3
Standard Deviation	18.9	16.9

Table 2: Token-level statistics of the constructed datasets. **Average Length** is the average number of tokens in the question and sentence-level answer text. **Query Coverage** is the percentage of tokens in the question that also appear in the sentence-level answer.

Question Type	SQuAD	NQ
what	47.7	15.3
who	9.6	32.6
how	9.3	5.0
when	6.2	20.3
which	5.5	2.0
where	3.8	13.1
why	1.4	0.6
other	16.5	11.1

Table 3: The distribution of question types in ReQA SQuAD and ReQA NQ. A question is assigned to a question type if it starts with the question type word. Note, types *what* and *which* include questions where a preposition (e.g. *at, by, in, on, with*) appears before the *wh-* word.

3.4 Discussion

A defining feature of the SQuAD dataset is that the questions are "back-written", with advance knowledge of the target answer and its surrounding context. One concern when adapting this data for a ReQA task is that questions may become ambiguous or underspecified when removed from the context of a specific document and paragraph. For example, SQuAD 1.1 contains the question "What instrument did he mostly compose for?". This question makes sense in the original context of the Wikipedia article on Frédéric Chopin, but is underspecified when asked in isolation, and could reasonably have other answers. One possible resolution would be to include the context title as part of the question context. However this is unrealistic from the point of view of end systems where the user doesn't have a specific document in mind.

This concern can be avoided by switching from "back-written" datasets to "web-search based" datasets. These include MS MARCO (Nguyen et al., 2016), TriviaQA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019). For these sets, questions are taken from natural sources, and a search engine is used in the process of constructing QA pairs.

However, there is an important caveat to mention when using web-search data to build ReQA tasks. In these datasets, the answers are derived from web documents retrieved by a search engine, where the question is used as the search query. This introduces a bias toward answers that are already retrievable through traditional search methods. By comparison, answers in SQuAD 1.1 may be found in "off-topic" documents, and it is valuable for an evaluation to measure the ability to retrieve such answers. Since both types of datasets (back-written and web-search based) have their advantages, we believe there is value in evaluating on ReQA tasks of both types.

4 Models and Results

In this section we evaluate neural models and classic information retrieval techniques on the ReQA SQuAD and ReQA NQ benchmark tasks.

4.1 Neural Baselines

Dual encoder models are learned functions that collocate queries and results in a shared embedding space. This architecture has shown strong performance on sentence-level retrieval tasks, including conversational response retrieval (Henderson et al., 2017; Yang et al., 2018), translation pair retrieval (Guo et al., 2018; Yang et al., 2019b) and similar text retrieval (Gillick et al., 2018). A dual encoder for use with ReQA has the schematic shape illustrated in Figure 3.

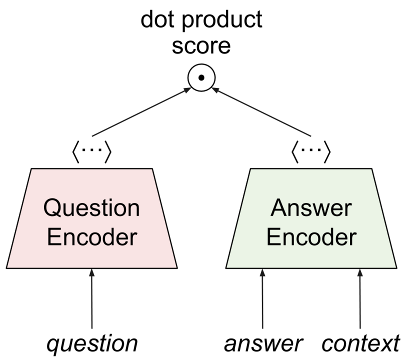

Figure 3: A schematic dual encoder for question-answer retrieval.

As our primary neural baseline, we take the recently released universal sentence encoder QA (USE-QA) model from Yang et al. (2019c)[10]. This is a multilingual QA retrieval model that co-trains a question-answer dual encoder along with secondary tasks of translation ranking and natural language inference. The model uses sub-word tokenization, with a 128k "sentencepiece" vocabulary (Kudo and Richardson, 2018). Question and answer text are encoded independently using a 6-layer transformer encoder (Vaswani et al., 2017), and then reduced to a fixed-length vector through average pooling. The final encoding dimensionality is 512. The training corpus contains over a billion question-answer pairs from popular online forums and QA websites like Reddit and StackOverflow.

As a second neural baseline, we include an internal QA model (QA_{Lite}) designed for use on mobile devices. Like USE-QA, this model is trained over online forum data, and uses a transformer-based text encoder. The core differences are reduction in width and depth of model layers, reduction of sub-word vocabulary size, and a decrease in the output embedding size from 512 dimensions to only 100.

Finally, we include the text embedding system InferSent, which, although not explicitly designed for question answering tasks, nevertheless produces strong results on a wide range of semantic tasks without requiring additional fine-tuning (Conneau et al., 2017). Note, however, that at 4096 dimensions, its embeddings are significantly larger than the other baselines presented. Other systems in this class include Skip-thought (Kiros et al., 2015), ELMo (Peters et al., 2018), and the Universal Sentence Encoder[11].

Table 4 presents the ReQA results for our baseline models. As expected, the larger USE-QA model outperforms the smaller QA_{Lite} model. The recall@1 score of 0.439 on ReQA SQuAD indicates that USE-QA is able to retrieve the correct answer from a pool of 91,707 candidates roughly 44% of the time. The ReQA NQ scores are lower, likely due to both the larger pool of candidate answers, as well as the lower degree of lexical overlap between questions and answers.

Table 5 illustrates the tradeoff between model accuracy and resource usage.

Model	**MRR**	**R@1**	**R@5**	**R@10**
ReQA SQuAD				
USE-QA	0.539	0.439	0.656	0.727
QA_{Lite}	0.412	0.325	0.507	0.576
InferSent	0.317	0.240	0.402	0.468
ReQA NQ				
USE-QA	0.234	0.147	0.317	0.391
QA_{Lite}	0.172	0.103	0.233	0.297
InferSent	0.080	0.043	0.109	0.145

Table 4: Mean reciprocal rank (MRR) and recall@K performance of neural baselines on ReQA SQuAD and ReQA NQ.

Model	**Size**	**Latency**[12]	**Memory**
	(MB)	(ms)	(MB)
USE-QA	392.9	17.3	71.8
QA_{Lite}	2.6	10.2	3.6

Table 5: Time and space tradeoffs of different models. Latency was measured on an Intel Xeon CPU E5-1650 v3 @ 3.50GHz, which has 6 cores and 12 threads.

[10]https://tfhub.dev/google/universal-sentence-encoder-multilingual-qa/1

[11]The non-QA versions of the Universal Sentence Encoder produce general semantic embeddings of text.

[12]This is the latency for encoding a single piece of text. However, by batching the encoding requests, it's possible to significantly reduce the amortized encoding time. In practice, batch sizes of 200 provide an amortized speedup of up to 5x.

4.2 BM25 Baseline

While neural retrieval systems are gaining popularity, TF-IDF based methods remain the dominant method for document retrieval, with the BM25 family of ranking functions providing a strong baseline (Robertson and Zaragoza, 2009). Unlike the neural models described above that can directly retrieve content at the sentence level, such methods generally consist of two stages: document retrieval, followed by sentence highlighting (Mitra and Craswell, 2018). Previous work in open domain question answering has shown that BM25 is a difficult baseline to beat when questions were written with advance knowledge of the answer (Lee et al., 2019).

To obtain our baseline using traditional IR methods, we constructed a paragraph-level retrieval task which allows a direct comparison between the neural systems in Table 4 and BM25.[13] We evaluate BM25 by measuring its ability to recall the paragraph containing the answer to the question.[14] To get a paragraph retrieval score for our neural baselines, we run sentence retrieval as before, and use the retrieved sentence to select the enclosing paragraph. As shown in Table 6, the USE-QA neural baseline outperforms BM25 on paragraph retrieval.

Model	MRR	R@1	R@5	R@10
ReQA SQuAD				
USE-QA	**0.634**	**0.533**	**0.756**	**0.823**
QA_{Lite}	0.503	0.407	0.613	0.689
InferSent	0.369	0.279	0.469	0.548
BM25[15]	0.602	0.517	0.702	0.755
ReQA NQ				
USE-QA	**0.366**	**0.247**	**0.486**	**0.578**
QA_{Lite}	0.274	0.177	0.366	0.450
InferSent	0.145	0.082	0.199	0.258
BM25	0.103	0.066	0.140	0.175

Table 6: Performance of various models on paragraph-level retrieval.

[13]We opted not to evaluate BM25 on sentence-level retrieval as earlier work has shown that traditional term-based document retrieval technologies are unsuccessful when applied to sentence-level retrieval (Allan et al., 2003).

[14]Our experiments make use of the implementation at `https://github.com/nhirakawa/BM25` with default hyperparameter settings.

[15]BM25 statistics were computed over the first 10,000 questions of each dataset, due to slow scoring speed.

5 Related Work

Open domain question answering is the problem of answering a question from a large collection of documents (Voorhees and Tice, 2000). Successful systems usually follow a two-step approach to answer a given question: first retrieve relevant articles or blocks, and then scan the returned text to identify the answer using a reading comprehension model (Jurafsky and Martin, 2018; Kratzwald and Feuerriegel, 2018; Yang et al., 2019a; Lee et al., 2019). While the reading comprehension step has been widely studied with many existing datasets (Rajpurkar et al., 2016; Nguyen et al., 2016; Dunn et al., 2017; Kwiatkowski et al., 2019), machine reading at scale is still a challenging task for the community.

Chen et al. (2017) recently proposed DrQA, treating Wikipedia as a knowledge base over which to answer factoid questions from SQuAD (Rajpurkar et al., 2016), CuratedTREC (Baudiš and Šedivý, 2015) and other sources. The task measures how well a system can successfully extract the answer span given a question, but it still relies on a document retrieval step. The ReQA eval differs from DrQA task by skipping the intermediate step and retrieving the answer sentence directly.

There is also a growing interest in answer selection at scale. Surdeanu et al. (2008) constructs a dataset with 142,627 question-answer pairs from Yahoo! Answers, with the goal of retrieving the right answer from all answers given a question. However, the dataset is limited to "how to" questions, which simplifies the problem by restricting it to a specific domain. Additionally the underlying data is not as broadly accessible as SQuAD and other more recent QA datasets, due to more restrictive terms of use.

WikiQA (Yang et al., 2015) is another task involving large-scale sentence-level answer selection. The candidate sentences are, however, limited to a small set of documents returned by Bing search, and is smaller than the scale of our ReQA tasks. WikiQA consists of 3,047 questions and 29,258 candidate answers, while ReQA SQuAD and ReQA NQ each contain over 20x that number of questions and over 3x that number of candidates (see Table 1). Moreover, as discussed in Section 3.4, restricting the domain of answers to top search engine results limits the evaluation's applicability for testing end-to-end retrieval.

Cakaloglu et al. (2018) made use of SQuAD for a retrieval task at the paragraph level. We extend this work by investigating sentence level retrieval and by providing strong sentence-level and paragraph-level baselines over a replicable construction of a retrieval evaluation set from the SQuAD data. Further, while Cakaloglu et al. (2018) trained their model on data drawn from SQuAD, we would like to highlight that our own strong baselines do not make use of any training data from SQuAD. We advocate for future work to attempt a similar approach of using sources of model training and evaluation data that are distinct as possible in order to provide a better picture of how well models generally perform a task.

Finally, Seo et al. (2018) construct a phrase-indexed question answering challenge that is similar to ReQA in requiring the question and the answer be encoded separately of one another. However, while ReQA focuses on sentence-based retrieval, their benchmark retrieves phrases, allowing for a direct F_1 and exact-match evaluation on SQuAD. Seo et al. (2019) demonstrate an implementation of a phrase-indexed question answering system using a combination of dense (neural) and sparse (term-frequency based) indices.

We believe that ReQA can help guide development of such systems by providing a point of evaluation between SQuAD, whose passages are too small to test retrieval performance, and SQuAD-Open (Chen et al., 2017), which operates at a realistic scale but is expensive and slow to evaluate. In practice, our evaluation runs completely in memory and finishes within two hours on a developer workstation, making it easy to integrate directly into the training process, where it can, for instance, trigger early stopping.

6 Conclusion

In this paper, we introduce Retrieval Question-Answering (ReQA) as a new benchmark for evaluating end-to-end answer retrieval models. The task assesses how well models are able to retrieve relevant sentence-level answers to queries from a large corpus. We describe a general method for converting reading comprehension QA tasks into cross-document answer retrieval tasks. Using SQuAD and Natural Questions as examples, we construct the ReQA SQuAD and ReQA NQ tasks, and evaluate several models on sentence- and paragraph-level answer retrieval. We find that a freely available neural baseline, USE-QA, outperforms a strong information retrieval baseline, BM25, on paragraph retrieval, suggesting that end-to-end answer retrieval can offer improvements over pipelined systems that first retrieve documents and then select answers within. We release our code for both evaluation and conversion of the datasets into ReQA tasks.

Acknowledgments

We thank our teammates from Descartes and other Google groups for their feedback and suggestions. We would like to recognize Javier Snaider, Igor Krivokon and Ray Kurzweil. Special thanks goes to Jonni Kanerva for developing the tailored sentence-breaking algorithm.

References

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 314–321, New York, NY, USA. ACM.

Petr Baudiš and Jan Šedivý. 2015. Modeling of the question answering task in the yodaqa system. In *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction - Volume 9283*, CLEF'15, pages 222–228, Berlin, Heidelberg. Springer-Verlag.

Tolgahan Cakaloglu, Christian Szegedy, and Xiaowei Xu. 2018. Text embeddings for retrieval from a large knowledge base. *CoRR*, abs/1810.10176.

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.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. 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 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.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *CoRR*, abs/1811.08008.

Mandy Guo, Qinlan Shen, Yinfei Yang, Heming Ge, Daniel Cer, Gustavo Hernandez Abrego, Keith Stevens, Noah Constant, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Effective parallel corpus mining using bilingual sentence embeddings. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 165–176, Belgium, Brussels. Association for Computational Linguistics.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. 2019. Training neural response selection for task-oriented dialogue systems. *arXiv preprint arXiv:1906.01543*.

Mandar Joshi, Eunsol Choi, Daniel S. 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*, Vancouver, Canada. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2018. *Speech and Language Processing (3rd Edition, in draft)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3294–3302, Cambridge, MA, USA. MIT Press.

Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive document retrieval for deep question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 576–581, Brussels, Belgium. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. 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*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

Bhaskar Mitra and Nick Craswell. 2018. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

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.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-08:*

HLT, pages 719–727, Columbus, Ohio. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*. Curran Associates, Inc.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 200–207, New York, NY, USA. ACM.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Yinfei Yang, Gustavo Hernández Ábrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019b. Improving multilingual sentence embedding using bidirectional dual encoder with additive margin softmax. *CoRR*, abs/1902.08564.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019c. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, Melbourne, Australia. Association for Computational Linguistics.