Dense Passage Retrieval for Open-Domain Question Answering

Vladimir Karpukhin*, Barlas Oğuz*, Sewon Min[†], Patrick Lewis,
Ledell Wu, Sergey Edunov, Danqi Chen[‡], Wen-tau Yih

Facebook AI   [†]University of Washington   [‡]Princeton University

{vladk, barlaso, plewis, ledell, edunov, scottyih}@fb.com
sewon@cs.washington.edu
danqic@cs.princeton.edu

Abstract

Open-domain question answering relies on efficient passage retrieval to select candidate contexts, where traditional sparse vector space models, such as TF-IDF or BM25, are the de facto method. In this work, we show that retrieval can be practically implemented using *dense* representations alone, where embeddings are learned from a small number of questions and passages by a simple dual-encoder framework. When evaluated on a wide range of open-domain QA datasets, our dense retriever outperforms a strong Lucene-BM25 system largely by 9%-19% absolute in terms of top-20 passage retrieval accuracy, and helps our end-to-end QA system establish new state-of-the-art on multiple open-domain QA benchmarks.[1]

1 Introduction

Open-domain textual question answering is a task that requires a system to answer factoid questions using a large collection of documents as the information source, without the need of pre-specifying topics or domains. This task not only has a clear value in information-seeking applications such as handling informational queries in natural language (Voorhees et al., 1999), but also poses a fundamental technical challenge in "machine reading at scale" (Chen et al., 2017).

Early QA systems are often complicated and consist of multiple components, covering sub-tasks like question analysis, document retrieval, passage retrieval, answer extraction and verification (Ferrucci (2012); Moldovan et al. (2003), *inter alia*). In contrast, the rise of deep learning and specifically the advances of reading comprehension (Rajpurkar et al., 2016) suggest a much simplified two-stage framework: (1) a context *retriever* first selects a small subset of passages where some of them contain the answer to the question, and then (2) a machine *reader* can thoroughly examine the retrieved contexts and identify the correct answer. While reducing open-domain QA to machine reading is a very reasonable strategy, a huge performance degradation is often observed in practice (Chen et al., 2017; Yang et al., 2019)[2], which suggests that the retriever needs to be enhanced and the reader also requires to be adjusted for multi-context input.

Mainstream approaches for building the retriever for open-domain QA use traditional information retrieval methods, including TF-IDF unigram/bigram matching (Chen et al., 2017) or BM25 (Robertson and Zaragoza, 2009) term weighting supported by tools like Lucene and Elastic Search (Wang et al., 2019; Yang et al., 2019; Nie et al., 2019). TF-IDF or BM25 can be viewed as representing the question and text as high-dimensional, sparse vectors (with weighting). These sparse vectors can be searched efficiently with an inverted index, and are effective in factoid question answering for which the retriever usually needs to narrow down the search space dramatically based on keywords. For instance, to find relevant passages for a question *"How many provinces did the Ottoman empire contain in the 17th century?"* or another question *"What part of the atom did Chadwick discover?"*, it is clear that looking up the text which contain the keyword *Ottoman* or *Chadwick* is crucial.

Conversely, the *dense*, latent semantic encoding of contexts and questions is *complementary to the sparse vector representation by design.* For example, synonyms or paraphrases that consist of completely different tokens may still be mapped to vectors close to each other. Consider the question *"Who is the bad guy in lord of the rings?"*, which can be answered from the context *"Sala Baker is*

---

*Equal contribution
[1]Our code will be available at https://fburl.com/qa-dpr.

[2]For instance, the exact match score on SQuAD v1.1 drops from above 80% to less than 40% (Yang et al., 2019).

an actor and stuntman from New Zealand. He is best known for portraying the villain Sauron in the Lord of the Rings trilogy ...". A term-based system would have difficulty retrieving such a context, while a dense retrieval system would be able to better match *"bad guy"* with *"villain"* and fetch the correct context. Dense encodings are also *learnable*. For instance, with pairs of questions and contexts that contain the answers, the distance between the question and context in the latent semantic space can effectively be optimized by learning the embedding function. This provides additional flexibility to have a task-specific representation. With special in-memory data structure and indexing schemes, search in the dense vector space can also be done efficiently, thanks to tools like FAISS (Johnson et al., 2017).

In this paper, we focus on improving open-domain question answering by replacing traditional information retrieval (IR) methods with dense representations for retrieval. Specifically, we learn an effective dense vector representation of questions and passages using a simple dual-encoder framework (Bromley et al., 1994), such that similarity in the corresponding latent semantic space yields a better retrieval function, compared to traditional sparse vector representations. Recent work made significant efforts in this direction by either treating the retrieval module as a latent variable (Lee et al., 2019; Guu et al., 2020) or applying similarity search directly at the phrase level (Seo et al., 2019). However, the former approach needs to adopt extra pre-training tasks to make the system perform well, while the latter approach demonstrates the importance of sparse vectors as removing it entirely made the system much worse. Instead, we demonstrate the feasibility of training a dense retrieval model by proper fine-tuning on a small number of question–passage pairs from existing datasets.

Our contributions are twofold. First, we demonstrate that with the proper training setup, simply fine-tuning the question and passage encoder on existing question-passage pairs is sufficient to greatly outperform a strong Lucene-BM25 system in terms of top-k retrieval accuracy. The dense representation is complementary to the sparse vector representation, and combining them can further improve the performance. Second, we verify that, in the context of open-domain question answering, a higher retrieval accuracy indeed translates to a higher end-to-end QA accuracy. As a result, we achieve new

state-of-the-art results on multiple QA datasets in the open-retrieval setting.

2 Background

The problem of open-domain textual question answering studied in this paper can be described as follows. Given a factoid question, such as *"Who first voiced Meg on Family Guy?"* or *"Where was the 8th Dalai Lama born?"*, a system is required to answer it using a large corpus of diversified topics. More specifically, we assume the extractive QA setting, in which the answer is restricted to a span appearing in one or more documents in the corpus. Let the corpus $\mathcal{C} = \{d_1, d_2, \cdots, d_D\}$, which consists of D documents. We first split each of the document into text passages of equal lengths as the basic retrieval units[3] and hence we get M passages in total $p_1, p_2, \ldots, p_M$ and each passage p_i can be viewed as a sequence of tokens $w_1^{(i)}, w_2^{(i)}, \cdots, w_{|p_i|}^{(i)}$. Given a question q, the task is to find a span $w_s^{(j)}, w_{s+1}^{(j)}, \cdots, w_e^{(j)}$ from one of the documents d_j that can answer the question. Notice that to cover a wide variety of domains, the corpus size can easily range from millions of documents (e.g., Wikipedia) to billions (e.g., the Web).

This form of question answering is also known as *machine reading at scale*, coined by Chen et al. (2017), due to the popular format of extractive reading comprehension. However, by scaling up from one document to a large collection of documents, open-domain question answering is faced with an additional technical challenge: *relevant context retrieval.* Although impressive progress has been made on reading comprehension, thanks mainly to the advance of deep neural networks, naively applying an existing *reader* model to all documents in the corpus is not practical, due to the expensive inference-time computation. As a result, any open-domain QA system needs to include an efficient *retriever* component that can select a small set of relevant texts, before applying the reader to extract the answer.[4] Formally speaking, a retriever $R : (q, \mathcal{C}) \rightarrow \mathcal{C}_\mathcal{F}$ is a function that takes as input a question q and a corpus $\mathcal{C}$ and returns a much

---

[3]The ideal size and boundary of a text passage are functions of both the retriever and reader. We also experimented with natural paragraphs in our preliminary trials and found that using fixed-length passages performs better in both retrieval and final QA accuracy, as consistent with the observations in (Wang et al., 2019)

[4]Exceptions include (Seo et al., 2019) and (Roberts et al., 2020), which *retrieves* and *generates* the answers, respectively.

smaller *filter set* of documents $\mathcal{C}_\mathcal{F} \subset \mathcal{C}$, where $|\mathcal{C}_\mathcal{F}| = k \ll |\mathcal{C}|$. For a fixed k, a *retriever* can be evaluated in isolation on *top-k retrieval accuracy*, which is the fraction of questions for which $\mathcal{C}_\mathcal{F}$ contains a span which answers the question.

3 Dense Passage Retriever (DPR)

We focus our research in this work on improving the *retrieval* component in open-domain question answering. Given a collection of M text passages, the goal of our dense retriever (DPR) is to index all the passages in a low-dimensional and continuous space, such that it can retrieve efficiently the top k passages relevant to the input question for the reader at run-time. Note that M can be very large (the corpus we used contains 21 million passages in our experiments as we will see in Section 4.1) and k is usually small, such as $20 \sim 100$. Below we give the design details of DPR.

3.1 Overview

Our dense passage retriever (DPR) uses a dense encoder $E_P(\cdot)$ which maps any text passage to a d-dimensional real-valued vectors and build an index for all the M passages we will use for retrieval. At run-time, DPR applies a different encoder $E_Q(\cdot)$ that maps the input question to a d-dimensional vector, and retrieves k passages of which vectors are the closest to the question vector. We define the similarity between the question and the passage using the dot product of the their vectors:

$$\text{sim}(q, p) = E_Q(q)^\intercal E_P(p), \quad (1)$$

Although more expressive model forms for measuring the similarity between a question and a passage do exist, such as networks consists of multiple layers of cross attentions, Eq. (1) is required to be decomposable so that the representations of the collection of passages can be pre-computed. Most decomposable similarity functions are some transformations of Euclidean distance (L2). For instance, cosine is equivalent to inner product for unit vectors and the Mahalanobis distance is equivalent to L2 distance in a transformed space. Inner product search has been widely used and studied, as well as its connection to cosine similarity and L2 distance (Mussmann and Ermon, 2016; Ram and Gray, 2012). In our preliminary experiments, we also tested cosine similarity but found no significant improvement. We thus choose the simpler inner product function and improve the dense passage retriever by learning better encoders.

Encoders Although in principle the question and passage encoders can be implemented by any neural network, in this work we use two independent BERT (Devlin et al., 2019) networks (base, uncased) and take the representation at the [CLS] token as the output, so $d = 768$.

Inference During inference time, we apply the passage encoder E_P to all the passages and index them using FAISS (Johnson et al., 2017) offline. FAISS is an extremely efficient, open-source library for similarity search and clustering of dense vectors, which can easily be applied to billions of vectors. Given a question q at run-time, we derive its embedding $v_q = E_Q(q)$ and retrieve the top k passages with embeddings closest to v_q.

3.2 Training

Training the encoders so that the dot-product similarity (Eq. (1)) becomes a good ranking function for retrieval is essentially a *metric learning* problem (Kulis, 2013). The goal is to create a vector space such that relevant pairs of questions and passages will have smaller distance (i.e., higher similarity) than the irrelevant pairs, by learning a better embedding.

Let $\mathcal{D} = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^- \rangle\}_{i=1}^m$ be the training data that consists of m instances. Each instance contains one question q_i and one relevant (positive) passage p_i^+, along with n irrelevant (negative) passages $p_{i,j}^-$. The loss function is the negative log likelihood of the correct passage:

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) \quad (2)$$
$$= -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}$$

Positive and negative passages For retrieval problems, it is often the case that positive examples are available explicitly, while negative examples need to be selected from an extremely large pool. For instance, relevant passages to a question may be given in a QA dataset, or can be found using the answer. All other passages in the collection, while not specified explicitly, can be viewed as irrelevant by default. In practice, how to select negative examples is often overlooked but could be decisive for learning a high-quality encoder. We generally consider three different types of negatives: (1) Random: any random passage from the corpus; (2) BM25: top passages returned by BM25 which don't contain the answer but match question

tokens heavily; (3) Gold: positive passages paired with other questions which appear in the training set. We will discuss the impact of different types of negative passages and training schemes in Section 5.4. Our best model uses gold passages from the same mini-batch and one BM25 negative passage. In particular, re-using gold passages from the same batch as negatives can make the computation efficient while achieving great performance. We will discuss this approach below.

In-batch negatives Assume that we have B questions in a mini-batch and each one is associated with a relevant passage. Let $\mathbf{Q}$ and $\mathbf{P}$ be the $(B \times d)$ matrix of question and passage embeddings in a batch of size B. $\mathbf{S} = \mathbf{Q}\mathbf{P}^T$ is a $(B \times B)$ matrix of similarity scores, where each row of which corresponds to a question, paired with B passages. In this way, we reuse computation and effectively train on B^2 (q_i, p_j) question/passage pairs in each batch. Any (q_i, p_j) pair is a positive example when $i = j$, and negative otherwise. This creates B training instances in each batch, where there are $B - 1$ negative passages for each question.

The trick of in-batch negatives has been used in the full batch setting (Yih et al., 2011) and more recently for mini-batch (Gillick et al., 2019; Henderson et al., 2017). This has been shown an effective strategy for learning a Siamese neural network (aka. dual encoder) model that boosts the number of training examples. As will be seen in Section 5, it is important in creating a high-quality passage retriever that helps advance the state of the art of open-domain question answering.

4   Experimental Setup

In this section, we describe the data we used for experiments and the basic setup.

4.1   Wikipedia Data Pre-processing

Following (Lee et al., 2019), we use the English Wikipedia dump from Dec. 20, 2018 as the source documents for answering questions. We first apply the pre-processing code released in DrQA (Chen et al., 2017) to extract the clean, text-portion of articles from the Wikipedia dump. This step removes semi-structured data, such as tables, infoboxes, lists, as well as the disambiguation pages. We then split each article into multiple, disjoint text blocks of 100 words as *passages*, serving as our basic retrieval units, following (Wang et al., 2019),

which results in 21,015,324 passages in the end.[5] Each passage is also prepended with the title of the Wikipedia article where the passage is from, along with an `[SEP]` token.

4.2   Question Answering Datasets

We use the same five QA datasets and training/dev/testing splitting method as in previous work (Lee et al., 2019). Below we briefly describe each dataset and refer readers to their paper for the details of data preparation.

Natural Questions (NQ) (Kwiatkowski et al., 2019) was designed for end-to-end question answering. The questions were mined from real Google search queries and the answers were spans in Wikipedia articles identified by annotators.

TriviaQA (Trivia) (Joshi et al., 2017) contains a set of trivia questions with answers that were originally scraped from the Web.

WebQuestions (WQ) (Berant et al., 2013) consists of questions selected using Google Suggest API, where the answers are entities in Freebase.

CuratedTREC (TREC) (Baudiš and Šedivỳ, 2015) sources questions from TREC QA tracks as well as various web sources and is intended for open-domain QA from unstructured corpora.

SQuAD v1.1 (Rajpurkar et al., 2016) is a popular benchmark dataset for reading comprehension. Annotators were presented with a Wikipedia paragraph, and asked to write questions that can be answered from the given text. Although SQuAD has been used previously for open-domain QA research, it is not ideal because many questions lack context in absence of the provided paragraph. We still include it in our experiments for providing a fair comparison to previous work and we will discuss more in Section 5.1.

Selecting positive passages Because only pairs of questions and answers are provided in TREC, WebQuestions and TriviaQA[6], we use the highest-ranked passage from BM25 that contains the answer as the positive passage. If none of the top 100 retrieved passages has the answer, the question will be discarded. For SQuAD and Natural Questions, since the original passages have been split and processed differently than our pool of candidate passages, we match and replace each gold passage

---

[5]However, Wang et al. (2019) also propose splitting documents into overlapping passages, which we do not find advantageous compared to the non-overlapping version.

[6]We used the unfiltered TriviaQA version and discarded the noisy evidence documents mined from Bing.

Dataset	Train		Dev	Test
Natural Questions	79,168	58,880	8,757	3,610
TriviaQA	78,785	60,413	8,837	11,313
WebQuestions	3,417	2,474	361	2,032
CuratedTREC	1,353	1,125	133	694
SQuAD	78,713	70,096	8,886	10,570

Table 1: Number of questions in each QA datasets. The second column of **Train** shows the numbers of questions after pruning. See text for detail.

with the corresponding passage in the candidate pool. We discard the questions when the matching is failed due to different Wikipedia versions or pre-processing. Table 1 shows the number of questions in training/dev/test sets for all the datasets and the actual questions used for training the retriever.

5 Experiments: Passage Retrieval

In this section, we evaluate the retrieval performance of our Dense Passage Retriever (DPR), along with analysis on how its output differs from traditional retrieval methods, the effects of different training schemes and the run-time efficiency.

The DPR model used in our main experiments is trained using the in-batch negative setting (Section 3.2) with a batch size of 128 and one additional BM25 negative passage per question. We trained the question and passage encoders for up to 30 epochs for large datasets (NQ, Trivia, SQuAD) and 100 epochs for small datasets (TREC, WQ), with a learning rate of 10^{-5} using Adam, linear scheduling with warm-up and dropout rate 0.1.

While it is good to have the flexibility to adapt the retriever to each dataset, it would also be desirable to obtain a single retriever that works well across the board. To this end, we train a *multi-dataset* encoder by combining training data from all datasets excluding SQuAD.[7] In addition to DPR, we also present the results of BM25, the traditional retrieval method[8] and BM25+DPR, using a linear combination of their scores as the new ranking function. Specifically, we re-rank top-2000 BM25 retrieved passages using $\mathrm{BM25}(q,p) + \lambda \cdot \mathrm{sim}(q,p)$ as the ranking function. We used $\lambda = 1.1$ based on the development set retrieval accuracy.

Figure 1: Retriever top-k accuracy with different number of training examples used in our dense passage retriever vs BM25. The results are measured on the development set of Natural Questions. Our DPR trained using 1,000 examples already outperforms BM25.

5.1 Main Results

Table 2 compares different passage retrieval systems on five QA datasets, using the top-k accuracy (k is 20 or 100). With the exception of SQuAD, DPR performs consistently better than BM25 on all datasets. The gap is especially large when k is small (e.g., 78.4% vs. 59.1% for top-20 accuracy on Natural Questions). When training with multiple datasets, TREC, the smallest dataset of the five, benefits greatly from more training examples. In contrast, Natural Questions and WebQuestions improve modestly and TriviaQA degrades slightly. Results can be improved further in some cases by combining DPR with BM25 in both single- and multi-dataset settings.

We conjecture that the lower performance on SQuAD is due to two reasons. First, the annotators wrote questions after seeing the passage. As a result, there is a high lexical overlap between passages and questions, which gives BM25 a clear advantage. Second, the data was collected from only 500+ Wikipedia articles and thus the distribution of training examples is extremely biased, as argued previously by Lee et al. (2019).

5.2 Sample Efficiency

We also explore how many training examples are needed to achieve good passage retrieval performance. Figure 1 illustrates the top-k retrieval accuracy with respect to different number of training examples, measured on the development set of Natural Questions. As is shown, a dense passage retriever trained using only 1,000 examples

---

[7]As discussed in Sec. 5.1, SQuAD is limited to a small set of Wikipedia documents, which introduces unwanted bias.

[8]Lucene implementation. BM25 parameters $b = 0.4$ (document length normalization) and $k_1 = 0.9$ (term frequency scaling) are tuned using development sets.
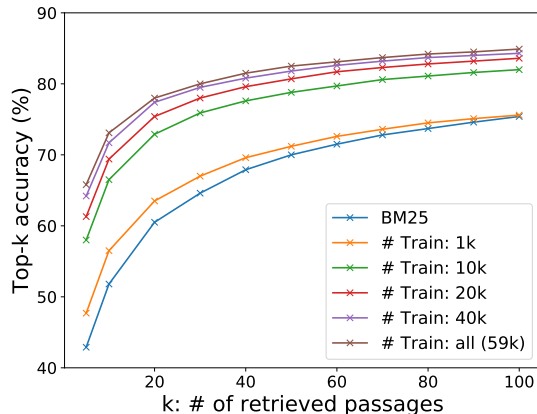
Training	Retriever	Top-20					Top-100				
		NQ	TriviaQA	WQ	TREC	SQuAD	NQ	TriviaQA	WQ	TREC	SQuAD
None	BM25	59.1	66.9	55.0	70.9	68.8	73.7	76.7	71.1	84.1	80.0
Single	DPR	78.4	79.4	73.2	79.8	63.2	85.4	**85.0**	81.4	89.1	77.2
	BM25 + DPR	76.6	79.8	71.0	85.2	**71.5**	83.8	84.5	80.5	92.7	**81.3**
Multi	DPR	**79.4**	78.8	**75.0**	**89.1**	51.6	**86.0**	84.7	**82.9**	93.9	67.6
	BM25 + DPR	78.0	**79.9**	74.7	88.5	66.2	83.9	84.4	82.3	**94.1**	78.6

Table 2: Top-20 & Top-100 retrieval accuracy on test sets, measured as the percentage of top 20/100 retrieved passages that contain the answer. *Single* and *Multi* denote that our Dense Passage Retriever (DPR) was trained using individial or combined training datasets (all the datasets excluding SQuAD). See text for more details.

Question	Passage received by BM25	Passage retrieved by DPR
What is the body of water between England and Ireland?	Title:British Cycling …**England** is not recognised as a region by the UCI, and there is no English cycling team outside the Commonwealth Games. For those occasions, British Cycling selects and supports the **England** team. Cycling is represented on the Isle of Man by the Isle of Man Cycling Association. Cycling in Northern **Ireland** is organised under Cycling Ulster, part of the all-Ireland governing **body** Cycling **Ireland**. Until 2006, a rival governing **body** existed, …	Title: Irish Sea …Annual traffic between Great Britain and **Ireland** amounts to over 12 million passengers and of traded goods. **The Irish Sea** is connected to the North Atlantic at both its northern and southern ends. To the north, the connection is through the North Channel between Scotland and Northern **Ireland** and the Malin Sea. The southern end is linked to the Atlantic through the St George's Channel between Ireland and Pembrokeshire, and the Celtic Sea. …
Who plays Thoros of Myr in Game of Thrones?	Title: No One (Game of Thrones) …He may be "no one," but there's still enough of a person left in him to respect, and admire who this girl is and what she's become. Arya finally tells us something that we've kind of known all along, that she's not no one, she's Arya Stark of Winterfell." "No One" saw the reintroduction of Richard Dormer and **Paul Kaye**, who portrayed Beric Dondarrion and **Thoros** of **Myr**, respectively, in the third season, …	Title: Pål Sverre Hagen Pål Sverre Valheim Hagen (born 6 November 1980) is a Norwegian stage and screen actor. He appeared in the Norwegian film "Max Manus" and played Thor Heyerdahl in the Oscar-nominated 2012 film "Kon-Tiki". Pl Hagen was born in Stavanger, Norway, the son of Roar Hagen, a Norwegian cartoonist who has long been associated with Norwayś largest daily, "VG". He lived in Jtten, a neighborhood in the city of Stavanger in south-western Norway. …

Table 3: Examples of passages returned from BM25 and DPR. Correct answers are written in **blue** and the content words in the question are written in bold.

already outperforms BM25, indicating that using a dense retriever to replace a term-matching retrieval system is feasible. Adding more training examples (from 1k to 59k) further improves the retrieval accuracy consistently.

5.3 Qualitative Analysis

While DPR performs better than BM25 in general, the retrieved passages of these two retrievers actually differ qualitatively. Methods like BM25 are sensitive to highly selective keywords and phrases, but cannot capture lexical variations or semantic relationships well. In contrast, DPR excels at semantic representation, but might lack sufficient capacity to represent salient phrases which appear rarely. Table 3 illustrates this phenomenon with two examples. In the first example, the top scoring passage from BM25 is irrelevant, even though keywords such as *England* and *Ireland* appear multiple times. In comparison, DPR is able to return the correct answer, presumably by matching *"body of water"* with semantic neighbors such as *sea* and *channel*, even though no lexical overlap exists. The second example is one where BM25 does better.

The salient phrase *"Thoros of Myr"* is critical, and DPR is unable to capture it.

5.4 Comparisons of Training Schemes

We experiment with different schemes of training DPR. Results on the development set of Natural Questions are summarized in Table 4. We first test the standard 1-of-N training setting (top block), where each question in the batch is paired with a positive passage and its own set of n negative passages (Eq. (2)). We found that using random, BM25 or gold passages (positive passages from other questions) perform similarly in this setting.

Next, we introduce in-batch negative training (Section 3.2). We find that using a similar setting (7 gold negative passages), in-batch negative training improved the results substantially. The key difference between the two is whether the gold negative passages come from the same batch or from the whole training set. Moreover, because in-batch negative training reuses other positive passages from the same batch, it is more memory-efficient and allows us to test bigger batch sizes. As a result, accuracy consistently increases as the batch size

Type	#N	IB	Top-5	Top-20	Top-100
Random	7	✗	47.0	64.3	77.8
BM25	7	✗	50.0	63.3	74.8
Gold	7	✗	42.6	63.1	78.3
Gold	7	✓	51.1	69.1	80.8
Gold	31	✓	52.1	70.8	82.1
Gold	127	✓	55.8	73.0	83.1
G.+BM25[(1)]	31	✓	65.0	77.3	84.4
G.+BM25[(2)]	31	✓	64.5	76.4	84.0
G.+BM25[(1)]	127	✓	**65.8**	**78.0**	**84.9**

Table 4: Comparison of different training schemes, measured as top-k retrieval accuracy on Natural Questions (development set). #N: number of negative examples, IB: in-batch training. G.+BM25[(1)] and G.+BM25[(2)] denote in-batch training with 1 or 2 additional BM25 negatives.

grows (middle block). Finally, we explore in-batch negative training with additional BM25 negative passages. We find that adding a single BM25 negative passage improved the result significantly while adding two did not help further.

5.5 Run-time Efficiency

The main reason that we require a retrieval component for open-domain QA is to reduce the number of candidate passages that the reader needs to consider, which is crucial for answering user's questions in real-time. We profiled the passage retrieval speed on a server with Intel Xeon CPU E5-2698 v4 @ 2.20GHz and 512GB memory. With the help of FAISS in-memory index for real-valued vectors[9], DPR can be made incredibly efficient, processing 995.0 questions per second, returning top 100 passages per question. In contrast, BM25/Lucene (implemented in Java, using file index) processes 23.7 questions per second per CPU thread.

On the other hand, the time required for building an index for dense vectors is much longer. Computing dense embeddings on 21-million passages is resource intensive, but can be easily parallelized, taking roughly 8.8 hours on 8 GPUs. However, building the FAISS index on 21-million vectors on a single server takes 8.5 hours. In comparison, building an inverted index using Lucene is much cheaper and takes only about 30 minutes in total.

---

[9]FAISS configuration: we used HNSW index type on CPU, neighbors to store per node = 512, construction time search depth = 200, search depth = 128.

6 Experiments: Question Answering

In this section, we experiment with how different passage retrievers affect the final QA accuracy.

6.1 End-to-end QA system

We implement an end-to-end question answering system in which we can plug our different retriever systems directly. Besides the retriever, our QA system consists of a neural *reader* that outputs the answer to the question. Given the top k retrieved passages (up to 100 in our experiments), the reader assigns a passage selection score to each passage. In addition, it extracts an answer span from each passage and assigns a span score. The best span from the passage with the highest passage selection score is chosen as the final answer. This approach has been shown to work well previously (Wang et al., 2019, 2018; Lin et al., 2018).

Specifically, let $\mathbf{P}_i \in \mathbb{R}^{L \times h}$ ($1 \leq i \leq k$) be a BERT (base, uncased in our experiments) representation for the i-th passage, where L is the maximum length of the passage and h the hidden dimension. The probabilities of a token being the starting/ending positions of an answer span and a passage being selected are defined as:

$$
\begin{aligned}
P_{\text{start},i}(s) &= \text{softmax}\big(\mathbf{P}_i \mathbf{w}_{\text{start}}\big)_s, \\
P_{\text{end},i}(t) &= \text{softmax}\big(\mathbf{P}_i \mathbf{w}_{\text{end}}\big)_t, \\
P_{\text{selected}}(i) &= \text{softmax}\big(\hat{\mathbf{P}}^\mathsf{T} \mathbf{w}_{\text{selected}}\big)_i,
\end{aligned}
$$

where $\hat{\mathbf{P}} = [\mathbf{P}_1^{[\text{CLS}]}, \dots, \mathbf{P}_k^{[\text{CLS}]}] \in \mathbb{R}^{h \times k}$ and $\mathbf{w}_{\text{start}}, \mathbf{w}_{\text{end}}, \mathbf{w}_{\text{selected}} \in \mathbb{R}^h$ are learnable vectors. We compute a span score of the s-th to t-th words from the i-th passage as $P_{\text{start},i}(s) \times P_{\text{end},i}(t)$, and a passage selection score of the i-th passage as $P_{\text{selected}}(i)$.

For training, we sample one positive and $m-1$ negative passages for each question at each iteration, where m is a hyperparamter (we use $m = 24$). The model training objective is maximum marginal likelihood for a passages could contain multiple correct answer spans. We use the batch size of 16 and 4 for large (NQ, Trivia, SQuAD) and small (TREC, WQ) datasets, respectively, and tune k on the development set. For experiments on small datasets under the *Multi* setting, in which using other datasets is allowed, we finetune the reader trained on Natural Questions to the target dataset. All experiments were done on eight 32GB GPUs.

Training	Model	NQ	TriviaQA	WQ	TREC	SQuAD
Single	BM25+BERT (Lee et al., 2019)	26.5	47.1	17.7	21.3	33.2
Single	ORQA (Lee et al., 2019)	33.3	45.0	36.4	30.1	20.2
Single	HardEM (Min et al., 2019a)	28.1	50.9	-	-	-
Single	GraphRetriever (Min et al., 2019b)	34.5	56.0	36.4	-	-
Single	PathRetriever (Asai et al., 2020)	32.6	-	-	-	**56.5**
Single	REALM$_{Wiki}$ (Guu et al., 2020)	39.2	-	40.2	46.8	-
Single	REALM$_{News}$ (Guu et al., 2020)	40.4	-	40.7	42.9	-
Single	BM25	32.6	52.4	29.9	24.9	38.1
	DPR	**41.5**	56.8	34.6	25.9	29.8
	BM25+DPR	39.0	57.0	35.2	28.0	36.7
Multi	DPR	**41.5**	56.8	**42.4**	49.4	24.1
	BM25+DPR	38.8	**57.9**	41.1	**50.6**	35.8

Table 5: End-to-end QA (Exact Match) Accuracy. REALM$_{Wiki}$ and REALM$_{News}$ are the same model but pretrained on Wikipedia and CC-News, respectively. *Single* and *Multi* denote that our Dense Passage Retriever (DPR) was trained using individual or combined training datasets (all except SQuAD). For WQ and TREC in the *Multi* setting, we finetune the reader trained on NQ.

6.2 Results

Table 5 summarizes our final end-to-end QA results, measured by *exact match* with the reference answer after minor normalization as in (Chen et al., 2017; Lee et al., 2019). From the table, we can see that higher retriever accuracy typically leads to better final QA results: in all cases except SQuAD, answers extracted from the passages retrieved by DPR are more likely to be correct, compared to those from BM25. For large datasets like NQ and TriviaQA, models trained using multiple datasets (Multi) perform comparably to those trained using the individual training set (Single). Conversely, on smaller datasets like WQ and TREC, the multi-dataset setting has a clear advantage. Overall, our DPR-based models outperform the previous state-of-the-art results on four out of the five datasets, with 1% to 12% absolute differences in exact match accuracy. It is interesting to contrast our results to those of ORQA (Lee et al., 2019) and also the concurrently developed approach, REALM (Guu et al., 2020). While both methods include additional pre-training tasks and employ an expensive end-to-end training regime, DPR manages to outperform them on both NQ and TriviaQA, simply by focusing on learning a strong passage retrieval model using pairs of questions and answers. The additional pre-training tasks are likely more useful only when the target training sets are small. Although the results of DPR on WQ and TREC in the single-dataset setting are less competitive, adding

more question–answer pairs helps boost the performance, achieving the new state of the art, as observed in the multi-dataset setting.

To compare our pipeline training approach with joint learning, we run an ablation on Natural Questions where the retriever and reader are jointly trained. Following Lee et al. (2019), the training procedure removes a passage scorer in the reader, and does not back-propagate to the passage encoder (E_P in Eq. (1)). This approach obtains a score of 36.8 EM, which suggests that our strategy of training a strong retriever and reader in isolation can leverage effectively available supervision, while outperforming a comparable joint training approach with a simpler design.

7 Related Work

Open-domain question answering, which requires a system to answer any factoid question using a large collection of documents, without pre-specifying a particular domain, has been popularized by the TREC QA tracks starting in the late '90s (Voorhees et al., 1999). Most QA systems in this era implemented a sophisticated pipeline approach or leveraged the Web redundancy to extract frequently occurring phrases as answers (Brill et al., 2002; Kwok et al., 2001). For Open-QA, retrieval has always been an important component. Not only does it effectively reduce the search space for answer extraction, but it also identifies the support context for the user to verify the answer. In practice, the sparse

vector space model like TF-IDF or BM25 is a very strong baseline for retrieval and has been viewed as the standard method applied broadly to various QA tasks (e.g., Chen et al., 2017; Nie et al., 2019; Min et al., 2019a; Wolfson et al., 2020). Augmenting text-based retrieval with externalstructured information, such as knowledge graph and Wikipedia hyper-links, has also been explored recently (Min et al., 2019b; Asai et al., 2020).

While the sparse vector space model has been the cornerstone of information retrieval, using continuous, dense vector representations of text has been proposed dating back to Latent Semantic Analysis (LSA) (Deerwester et al., 1990). Such approaches complement the sparse vector methods as they can potentially give high similarity scores to text pairs that are semantically relevant, even without exact token matching. The dense representation alone, however, is typically inferior to the sparse representation. In contrast to unsupervised dimensionality reduction methods (e.g., LSA), leveraging labeled data to train dense encoders for embedding both documents and queries has become popular recently (Yih et al., 2011; Huang et al., 2013; Gillick et al., 2019; Humeau et al., 2019). To ensure that the embedding can be pre-computed and indexed offline, the basic architecture follows the Siamese neural network design (Bromley et al., 1994; Chopra et al., 2005). The interaction between document and query will only occur on the final top-level, through standard distance functions such as inner-product or cosine similarity. When the training methods can use the labeled pairs to effectively generate a large number of examples, such as the in-batch random negative pairs (Yih et al., 2011; Henderson et al., 2017), or when there exist a big collection of positive and negative pairs (Huang et al., 2013), dense representation can outperform sparse representation. Enabled by the modern neural network toolkits, researchers have started to explore an end-to-end training scheme that includes pre-training, retrieval and also answer extraction jointly (Lee et al., 2019; Guu et al., 2020). While the initial results are positive, the overall framework becomes inevitably more complex. In contrast, our model provides a simple and yet effective solution that shows strong empirical performance.

8 Conclusion

In this work, we demonstrated that dense embedding-based retrieval can outperform and po-

tentially replace the traditional sparse retrieval component in open-domain question answering. We showed that while a simple approach can be made to work surprisingly well, there are some critical ingredients to training a dense retriever successfully. Dense and sparse retrieval are found to be largely complementary and can be combined for best accuracy. As a result of improved retrieval performance, we obtained new state-of-the-art results on multiple open-domain question answering benchmarks.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over Wikipedia graph for question answering. In *ICLR*.

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the askmsr question-answering system. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 257–264.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "Siamese" time delay neural network. In *NIPS*, pages 737–744.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*, pages 1870–1879.

Sumit Chopra, Raia Hadsell, Yann LeCun, et al. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Conference on computer vision and pattern recognition (CVPR)*.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

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

David A Ferrucci. 2012. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1–1.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *CoNLL*.

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

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yunhsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for Web search using clickthrough data. In *CIKM*, pages 2333–2338.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Real-time inference in multi-sentence tasks with deep pretrained transformers. *CoRR*, abs/1905.01969.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, pages 1601–1611.

Brian Kulis. 2013. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

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. *TACL*.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the Web. *ACM Trans. Inf. Syst.*, 19(3):242–262.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Association for Computational Linguistics (ACL)*, pages 6086–6096.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Association for Computational Linguistics (ACL)*, pages 1736–1745.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019a. A discrete hard EM approach for weakly supervised question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154.

Stephen Mussmann and Stefano Ermon. 2016. Learning and inference via maximum inner product search. In *International Conference on Machine Learning*, pages 2587–2596.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 931–939.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

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 *ACL*.

Ellen M Voorhees et al. 1999. The TREC-8 question answering track report. In *Trec*, volume 99, pages 77–82.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

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

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *arXiv preprint arXiv:2001.11770*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. In *HLT-NAACL*, pages 72–77.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Computational Natural Language Learning (CoNLL)*, pages 247–256.