Abstract

Language model pre-training has been shown to capture a surprising amount of world knowledge, crucial for NLP tasks such as question answering. However, this knowledge is stored implicitly in the parameters of a neural network, requiring ever-larger networks to cover more facts.

To capture knowledge in a more modular and interpretable way, we augment language model pre-training with a latent knowledge retriever, which allows the model to retrieve and attend over documents from a large corpus such as Wikipedia, used during pre-training, fine-tuning and inference. For the first time, we show how to pre-train such a knowledge retriever in an unsupervised manner, using masked language modeling as the learning signal and backpropagating through a retrieval step that considers millions of documents.

We demonstrate the effectiveness of Retrieval-Augmented Language Model pre-training (REALM) by fine-tuning on the challenging task of Open-domain Question Answering (Open-QA). We compare against state-of-the-art models for both explicit and implicit knowledge storage on three popular Open-QA benchmarks, and find that we outperform all previous methods by a significant margin (4–16% absolute accuracy), while also providing qualitative benefits such as interpretability and modularity.

1. Introduction

Recent advances in language model pre-training have shown that models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2019) store a surprising amount of world knowledge, acquired from the massive text corpora they are trained on (Petroni et al., 2019). For example, BERT is able to correctly predict the missing word in the following sentence: “The ____ is the currency of the United Kingdom” (answer: “pound”).

In these language models, the learned world knowledge is stored implicitly in the parameters of the underlying neural network. This makes it difficult to determine what knowledge is stored in the network and where. Furthermore, storage space is limited by the size of the network—to capture more world knowledge, one must train ever-larger networks, which can be prohibitively slow or expensive.

To capture knowledge in a more interpretable and modular way, we propose a novel framework, Retrieval-Augmented Language Model (REALM) pre-training, which augments...
REALM: Retrieval-Augmented Language Model Pre-Training

language model pre-training algorithms with a learned textual knowledge retriever. In contrast to models that store knowledge in their parameters, this approach explicitly exposes the role of world knowledge by asking the model to decide what knowledge to retrieve and use during inference. Before making each prediction, the language model uses the retriever to retrieve documents1 from a large corpus such as Wikipedia, and then attends over those documents to help inform its prediction. Learning this model end-to-end requires backpropagating through a retrieval step that considers an entire corpus of textual knowledge, as shown in Figure 1.

The key intuition of REALM is to train the retriever using a performance-based signal from unsupervised text: a retrieval that improves the language model’s perplexity is helpful and should be rewarded, while an uninformative retrieval should be penalized. For example, in Figure 1, if the model needs to fill the blank in “the ___ at the top of the pyramid”, the retriever should be rewarded for selecting a document containing “The pyramidion on top allows for less material higher up the pyramid”. We achieve this behavior by modeling our retrieve-then-predict approach as a latent variable language model and optimizing the marginal likelihood.

Incorporating a large-scale neural retrieval module during pre-training constitutes a significant computational challenge, since the retriever must consider millions of candidate documents for each pre-training step, and we must backpropagate through its decisions. To address this, we structure the retriever such that the computation performed for each document can be cached and asynchronously updated, and selection of the best documents can be formulated as Maximum Inner Product Search (MIPS).

Numerous prior works have demonstrated the benefit of adding a discrete retrieval step to neural networks (Miller et al., 2016; Chen et al., 2017), but did not apply the framework to language model pre-training and employed non-learned retrievers to handle large-scale document collections. In the language modeling literature, the k-Nearest Neighbor Language Model (Khandelwal et al., 2019) (kNN-LM) retrieves similar LM examples to improve memorization. However, kNN-LM was not fine-tuned for downstream tasks, perhaps because it is unclear how to adapt the retrieval mechanism: a kNN can only use examples labeled for the target task—during fine-tuning, this precludes LM examples, which contain the desired world knowledge. In contrast, REALM’s retriever is designed to transfer to other tasks, and the retrieval is just text, not a labeled example.

We evaluate our approach by fine-tuning the models pre-trained with REALM on the task of Open-domain Question Answering (Open-QA), one of the most knowledge-intensive tasks in natural language processing. We evaluate on three popular Open-QA benchmarks (NaturalQuestions-Open, WebQuestions, and CuratedTREC) and compare to state-of-the-art Open-QA models, including both extremely large models that store knowledge implicitly (such as T5) as well as previous approaches that also use a knowledge retriever to access external knowledge, but implement retrieval in a more heuristic fashion (Lee et al., 2019; Min et al., 2019a; Asai et al., 2019). REALM achieves new state-of-the-art results on all three benchmarks, significantly outperforming all previous systems by 4-16% absolute accuracy. We also demonstrate qualitative benefits of REALM, including interpretability and modularity.

2. Background

Language model pre-training The goal of language model pre-training is to learn useful representations of language, usually from unlabeled text corpora. The resulting pre-trained model can then be further trained (fine-tuned) for a downstream task of primary interest (in our case, Open-QA), often leading to better generalization than training from scratch (Dai & Le, 2015; Radford et al., 2019).

We focus on the masked language model2 (MLM) variant of pre-training popularized by BERT (Devlin et al., 2018). In its basic form, an MLM is trained to predict the missing tokens in an input text passage. Given an unlabeled pre-training corpus \( \mathcal{X} \) (e.g., Wikipedia text), a training example \((x, y)\) can be generated by randomly masking tokens in a sampled piece of text (e.g., \(x = \text{“The [MASK] is the currency [MASK] the UK”}; \ y = \text{(“pound”, “of”)}\)). The model uses its representation of the masked input \( x \) to predict the token that should go in each mask. A good MLM must learn to encode syntactic and semantic information (e.g., to predict “of”) as well as some world knowledge (e.g., to predict “pound”).

Open-domain question answering (Open-QA) To measure a model’s ability to incorporate world knowledge, we need a downstream task where world knowledge is critical. Perhaps one of the most knowledge-intensive tasks in natural language processing is open-domain question answering (Open-QA): given a question \( x \) such as “What is the currency of the UK?”, a model must output the correct answer string \( y \), “pound”. The “open” part of Open-QA refers to the fact that the model does not receive a pre-identified document that is known to contain the answer, unlike traditional reading comprehension (RC) tasks.

1We use the term “document” loosely to refer to a passage from the knowledge corpus, not necessarily a whole article.

2Strictly speaking, MLM is not a standard language model, since it does not define a distribution over the entire sequence of tokens. In the paper we sometimes abuse the term “language model” slightly to make the phrase shorter.
We focus on Open-QA systems that utilize a textual knowledge corpus as the knowledge source. Many of these systems employ a retrieval-based approach: given a question \( x \), retrieve potentially relevant documents \( z \) from the corpus \( Z \), and then extract an answer \( y \) from the documents (Brill et al., 2002; Chen et al., 2017; Lee et al., 2019).

Our approach, REALM, is inspired by this paradigm and extends it to language model pre-training. Alternatively, some recent work has proposed generation-based systems that apply a sequence-to-sequence model on \( x \) to directly generate \( y \) token-by-token (Lewis et al., 2019; Raffel et al., 2019). We will compare against state-of-the-art systems from both paradigms in our experiments.

3. Approach

We start by formalizing REALM’s pre-training and fine-tuning tasks as a retrieve-then-predict generative process in Section 3.1. Then in Section 3.2, we describe the model architectures for each component of that process. In Section 3.3, we show how to implement REALM pre-training and fine-tuning by maximizing the likelihood of REALM’s generative process. En route, we address important computational challenges, explain why training works, and also derive strategies for injecting useful inductive biases. The overall framework is illustrated in Figure 2.

3.1. REALM’s generative process

For both pre-training and fine-tuning, REALM takes some input \( x \) and learns a distribution \( p(y \mid x) \) over possible outputs \( y \). For pre-training, the task is masked language modeling: \( x \) is a sentence from a pre-training corpus \( \mathcal{X} \) with some tokens masked out, and the model must predict the value of those missing tokens, \( y \). For fine-tuning, the task is Open-QA: \( x \) is a question, and \( y \) is the answer.

REALM decomposes \( p(y \mid x) \) into two steps: retrieve, then predict. Given an input \( x \), we first retrieve possibly helpful documents \( z \) from a knowledge corpus \( Z \). We model this as a sample from the distribution \( p(z \mid x) \). Then, we condition on both the retrieved \( z \) and the original input \( x \) to generate the output \( y \)—modeled as \( p(y \mid z, x) \). To obtain the overall likelihood of generating \( y \), we treat \( z \) as a latent variable and marginalize over all possible documents \( z \), yielding

\[
p(y \mid x) = \sum_{z \in Z} p(y \mid z, x) p(z \mid x).
\]  

3.2. Model architecture

We now describe the two key components: the neural knowledge retriever, which models \( p(z \mid x) \), and the knowledge-augmented encoder, which models \( p(y \mid z, x) \).

Knowledge Retriever The retriever is defined using a dense inner product model:

\[
p(z \mid x) = \frac{\exp f(x, z)}{\sum_{z'} \exp f(x, z')},
\]

\[
f(x, z) = \text{Embed}_{\text{input}}(x)^\top \text{Embed}_{\text{doc}}(z),
\]

where \( \text{Embed}_{\text{input}} \) and \( \text{Embed}_{\text{doc}} \) are embedding functions that map \( x \) and \( z \) respectively to \( d \)-dimensional vectors. The relevance score \( f(x, z) \) between \( x \) and \( z \) is defined as the inner product of the vector embeddings. The retrieval distribution is the softmax over all relevance scores.

We implement the embedding functions using BERT-style Transformers (Devlin et al., 2018). Following standard practices, we join spans of text by applying wordpiece tokenization, separating them with \([\text{SEP}] \) tokens, prefixing a \([\text{CLS}] \) token, and appending a final \([\text{SEP}] \) token.

As in Devlin et al. (2018), we pass this into a Transformer, which produces one vector for each token, including the vector corresponding to \([\text{CLS}] \) which is used as a “pooled” representation of the sequence (denoted \( BERT_{\text{CLS}} \)). Finally, we perform a linear projection to reduce the dimensionality of the vector, denoted as a projection matrix \( W \):

\[
\text{Embed}_{\text{input}}(x) = W_{\text{input}} BERT_{\text{CLS}}(\text{join}_{\text{BERT}}(x))
\]

\[
\text{Embed}_{\text{doc}}(z) = W_{\text{doc}} BERT_{\text{CLS}}(\text{join}_{\text{BERT}}(z_{\text{title}}, z_{\text{body}}))
\]

where \( z_{\text{title}} \) is the document’s title and \( z_{\text{body}} \) is its body. We let \( \theta \) denote all parameters associated with the retriever, which include the Transformer and projection matrices.

Knowledge-Augmented Encoder Given an input \( x \) and a retrieved document \( z \), the knowledge-augmented encoder defines \( p(y \mid z, x) \). We join \( x \) and \( z \) into a single sequence that we feed into a Transformer (distinct from the one used in the retriever). This allows us to perform rich cross-attention between \( x \) and \( z \) before predicting \( y \). See Figure 1 for a concrete example.

At this stage, the architectures for pre-training and fine-tuning differ slightly. For the masked language model pre-training task, we must predict the original value of each \([\text{MASK}] \) token in \( x \). To do so, we use the same masked
language modeling (MLM) loss as in Devlin et al. (2018):
\[
p(y | z, x) = \prod_{j=1}^{J_z} p(y_j | z, x)
\]
\[
p(y_j | z, x) \propto \exp(w_j^T \text{BERT}_{\text{MASK}}(j) \text{join}_{\text{BERT}}(x, z_{\text{body}}))
\]
where BERT_{\text{MASK}}(j) denotes the Transformer output vector corresponding to the \(j\)th masked token, \(J_z\) is the total number of [MASK] tokens in \(x\), and \(w_j\) is a learned word embedding for token \(y_j\).

For Open-QA fine-tuning, we wish to produce the answer string \(y\). Following previous reading comprehension work (Rajpurkar et al., 2016; Seo et al., 2016; Lee et al., 2016; Clark & Gardner, 2017), we will assume that the answer \(y\) can be found as a contiguous sequence of tokens in some document \(z\). Let \(S(z, y)\) be the set of spans matching \(y\) in \(z\). Then we can define \(p(y | z, x)\) as:

\[
p(y | z, x) \propto \sum_{s \in S(z, y)} \exp(\text{MLP}([h_{\text{START}(s)}; h_{\text{END}(s)}]))
\]
\[
h_{\text{START}(s)} = \text{BERT}_{\text{START}}(s) \text{join}_{\text{BERT}}(x, z_{\text{body}}),
\]
\[
h_{\text{END}(s)} = \text{BERT}_{\text{END}}(s) \text{join}_{\text{BERT}}(x, z_{\text{body}}),
\]
where BERT_{\text{START}}(s) and BERT_{\text{END}}(s) denote the Transformer output vectors corresponding to the start and end tokens of span \(s\), respectively, while MLP denotes a feed-forward neural network. We will let \(\phi\) denote all parameters associated with the knowledge-augmented encoder.

**3.3. Training**

For both pre-training and fine-tuning, we train by maximizing the log-likelihood \(\log p(y | x)\) of the correct output \(y\). Since both the knowledge retriever and knowledge-augmented encoder are differentiable neural networks, we can compute the gradient of \(\log p(y | x)\) (defined in Equation 1) with respect to the model parameters \(\theta\) and \(\phi\), and optimize using stochastic gradient descent.

The key computational challenge is that the marginal probability \(p(y | x) = \sum_{z \in Z} p(y | x, z) p(z | x)\) involves a summation over all documents \(z\) in the knowledge corpus \(Z\). We approximate this by instead summing over the top \(k\) documents with highest probability under \(p(z | x)\)—this is reasonable if most documents have near zero probability.

Even with this approximation, we still need an efficient way to find the top \(k\) documents. Note that the ordering of documents under \(p(z | x)\) is the same as under the relevance score \(f(x, z) = \text{Embed}_{\text{input}}(x)^T \text{Embed}_{\text{doc}}(z)\), which is an inner product. Thus, we can employ Maximum Inner Product Search (MIPS) algorithms to find the approximate top \(k\) documents, using running time and storage space that scale sub-linearly with the number of documents (Ram & Gray, 2012; Shrivastava & Li, 2014; Shen et al., 2015).

To employ MIPS, we must pre-compute \(\text{Embed}_{\text{doc}}(z)\) for every \(z \in Z\) and construct an efficient search index over these embeddings. However, this data structure will no longer be consistent with \(p(z | x)\) if the parameters \(\theta\) of \(\text{Embed}_{\text{doc}}\) are later updated. Hence, the search index goes “stale” after every gradient update on \(\theta\).

Our solution is to “refresh” the index by asynchronously re-embedding and re-indexing all documents every several hundred training steps. The MIPS index is slightly stale between refreshes, but note that it is only used to select the top \(k\) documents. We recompute \(p(z | x)\) and its gradient, using the fresh \(\theta\), for these top \(k\) documents after retrieving them. In Section 4.5, we empirically demonstrate that this procedure results in stable optimization, provided that refreshes happen at a sufficiently frequent rate.

**Implementing asynchronous MIPS refreshes** We asynchronously refresh the MIPS index by running two jobs in parallel: a primary trainer job, which performs gradient updates on the parameters, and a secondary index builder job, which embeds and indexes the documents. As shown...
We now evaluate our approach on the Open-QA task. In this section, we describe in detail the benchmarks used and the different approaches to which we compare empirically.

### 4. Experiments

We now evaluate our approach on the Open-QA task. In this section, we describe in detail the benchmarks used and the different approaches to which we compare empirically.
4.1. Open-QA Benchmarks

A number of benchmarks have been proposed for Open-QA. In this work, we focus on datasets where the question writers did not already know the answer. This yields questions that reflect more realistic information-seeking needs, and also avoids artifacts that can arise if the question is formulated with a particular answer in mind. A deeper justification is given in Lee et al. (2019). In all cases, the predicted answer is evaluated via exact match with any reference answer, following previous Open-QA work (Chen et al., 2017).

NaturalQuestions-Open The NaturalQuestions dataset (Kwiatkowski et al., 2019) consists of naturally occurring Google queries and their answers. Each answer also comes with an “answer type”: following Lee et al. (2019), we only keep questions that are categorized as “short answer type” with at most five tokens. The dataset also provides a suggested Wikipedia document to retrieve; like all models we compare against, we do not provide this to our model.

WebQuestions The WebQuestions dataset (Berant et al., 2013) was collected from the Google Suggest API, using one seed question and expanding the set to related questions. We follow the setting defined by Chen et al. (2017).

CuratedTrec The CuratedTrec dataset is a collection of question-answer pairs drawn from real user queries issued on sites such as MSNSearch and AskJeeves. To account for multiple correct answers or different spelling variations, the answers in this dataset are defined as regular expressions that match all correct answers. It is unclear how to train generation-based models with this type of supervision, so we do not evaluate them on this dataset.

4.2. Approaches compared

Retrieval-based Open-QA Most existing Open-QA systems answer the input question by first retrieving potentially relevant documents from a knowledge corpus, and then using a reading comprehension system to extract an answer from the documents. In this paradigm, the knowledge is stored explicitly in the corpus. We wish to compare different methods for implementing retrieval.

Many approaches use non-learned heuristic retrieval such as sparse bag-of-words matching (Robertson et al., 2009) or entity linking on the question to select a small set of relevant documents (e.g., 20). These documents are typically then re-ranked using a learned model, but coverage may be limited by the initial heuristic retrieval step. Approaches such as DrQA (Chen et al., 2017), HardEM (Min et al., 2019a), GraphRetriever (Min et al., 2019b), and PathRetriever (Asai et al., 2019) in Table 1 are in this category.

Some recent approaches have proposed to implement learnable retrieval using a MIPS index. ORQA (Lee et al., 2019) formulates Open-QA using a similar latent variable model as REALM, and also trains by maximizing the marginal likelihood. However, REALM adds a novel language model pre-training step, and backpropagates into the MIPS index, rather than using a fixed index. In Table 1, we directly compare the two. It is also important to note that the retrievers for both REALM pretraining and ORQA are initialized using the Inverse Cloze Task, described in Section 3.4.

Generation-based Open-QA An emerging alternative approach to Open-QA is to model it as a sequence prediction task: simply encode the question, and then decode the answer token-by-token based on the encoding. While it was initially unclear how large amounts of knowledge could be injected into the model, GPT-2 (Radford et al., 2019) hinted at the possibility of directly generating answers without using any given context via sequence-to-sequence. However, their performance was not competitive possibly due to the lack of fine-tuning. Orthogonally, T5 (Raffel et al., 2019) showed that directly generating answers without explicit extraction from the given context is viable approach, but they only experimented on the reading comprehension task, where a context document is provided.

For the most competitive and comparable generation-based baseline, we compare to concurrent work which fine-tunes T5 for Open-QA (Roberts et al., 2020). We compare against the Base, Large, and even larger 11-billion parameter model to measure the effect of model size.

4.3. Implementation Details

Pre-training We pre-train for 200k steps on 64 Google Cloud TPUs, with a batch size of 512 and a learning rate of 3e-5, using BERT’s default optimizer. The document embedding step for the MIPS index is parallelized over 16 TPUs. For each example, we retrieve and marginalize over 8 candidate documents, including the null document \( \emptyset \).

We experiment with two choices of the pre-training corpus \( \mathcal{X} \): (1) Wikipedia, which is identical to the knowledge corpus \( \mathcal{Z} \), and (2) CC-News, our reproduction of the corpus of English news proposed by Liu et al. (2019).

Fine-tuning We follow the ORQA (Lee et al., 2019) fine-tuning approach but initialized with the pre-trained REALM components. The knowledge corpus is derived from the December 20, 2018 snapshot of English Wikipedia. Documents are greedily split into chunks of up to 288 BERT wordpieces, resulting in just over 13 million retrieval candi-

---

4 We initially conducted our own T5 experiments using the code from https://tinyurl.com/t5-openqa-colab (Raffel et al., 2019). We now report results from the concurrent work of Roberts et al. (2020), which has an improved fine-tuning procedure.
Table 1. Test results on Open-QA benchmarks. The number of train/test examples are shown in parentheses below each benchmark. Predictions are evaluated with exact match against any reference answer. Sparse retrieval denotes methods that use sparse features such as TF-IDF and BM25. Our model, REALM, outperforms all existing systems.

<table>
<thead>
<tr>
<th>Name</th>
<th>Architectures</th>
<th>Pre-training</th>
<th>NQ  (79k/4k)</th>
<th>WQ  (3k/2k)</th>
<th>CT  (1k/1k)</th>
<th># params</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-Baseline (Lee et al., 2019)</td>
<td>Sparse Retr.+Transformer</td>
<td>BERT</td>
<td>26.5</td>
<td>17.7</td>
<td>21.3</td>
<td>110m</td>
</tr>
<tr>
<td>T5 (base) (Roberts et al., 2020)</td>
<td>Transformer Seq2Seq</td>
<td>T5 (Multitask)</td>
<td>27.0</td>
<td>29.1</td>
<td>-</td>
<td>223m</td>
</tr>
<tr>
<td>T5 (large) (Roberts et al., 2020)</td>
<td>Transformer Seq2Seq</td>
<td>T5 (Multitask)</td>
<td>29.8</td>
<td>32.2</td>
<td>-</td>
<td>738m</td>
</tr>
<tr>
<td>T5 (11b) (Roberts et al., 2020)</td>
<td>Transformer Seq2Seq</td>
<td>T5 (Multitask)</td>
<td>34.5</td>
<td>37.4</td>
<td>-</td>
<td>11318m</td>
</tr>
<tr>
<td>DrQA (Chen et al., 2017)</td>
<td>Sparse Retr.+DocReader</td>
<td>N/A</td>
<td>-</td>
<td>20.7</td>
<td>25.7</td>
<td>34m</td>
</tr>
<tr>
<td>HardEM (Min et al., 2019a)</td>
<td>Sparse Retr.+Transformer</td>
<td>BERT</td>
<td>28.1</td>
<td>-</td>
<td>-</td>
<td>110m</td>
</tr>
<tr>
<td>GraphRetriever (Min et al., 2019b)</td>
<td>GraphRetriever+Transformer</td>
<td>BERT</td>
<td>31.8</td>
<td>31.6</td>
<td>-</td>
<td>110m</td>
</tr>
<tr>
<td>PathRetriever (Asai et al., 2019)</td>
<td>PathRetriever+Transformer</td>
<td>MLM</td>
<td>32.6</td>
<td>-</td>
<td>-</td>
<td>110m</td>
</tr>
<tr>
<td>ORQA (Lee et al., 2019)</td>
<td>Dense Retr.+Transformer</td>
<td>ICT+BERT</td>
<td>33.3</td>
<td>36.4</td>
<td>30.1</td>
<td>330m</td>
</tr>
<tr>
<td>ORQA (more fine-tune epochs)</td>
<td>Dense Retr.+Transformer</td>
<td>ICT+BERT</td>
<td>34.8</td>
<td>35.4</td>
<td>28.7</td>
<td>330m</td>
</tr>
<tr>
<td>Ours (X = Wikipedia, Z = Wikipedia)</td>
<td>Dense Retr.+Transformer</td>
<td>REALM</td>
<td>39.2</td>
<td>40.2</td>
<td>46.8</td>
<td>330m</td>
</tr>
<tr>
<td>Ours (X = CC-News, Z = Wikipedia)</td>
<td>Dense Retr.+Transformer</td>
<td>REALM</td>
<td>40.4</td>
<td>40.7</td>
<td>42.9</td>
<td>330m</td>
</tr>
</tbody>
</table>

Table 2. Ablation experiments on NQ’s development set. Unless otherwise stated, the pre-training corpus $X$ is Wikipedia.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Exact Match</th>
<th>Zero-shot Retrieval</th>
<th>Recall@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (X = CC-News)</td>
<td>38.5</td>
<td>52.0</td>
<td></td>
</tr>
<tr>
<td>REALM</td>
<td>38.2</td>
<td>38.5</td>
<td></td>
</tr>
<tr>
<td>REALM retriever+Baseline encoder</td>
<td>37.4</td>
<td>38.5</td>
<td></td>
</tr>
<tr>
<td>Baseline retriever+REALM encoder</td>
<td>35.3</td>
<td>13.9</td>
<td></td>
</tr>
<tr>
<td>Baseline (ORQA)</td>
<td>31.3</td>
<td>13.9</td>
<td></td>
</tr>
<tr>
<td>REALM with random uniform masks</td>
<td>32.3</td>
<td>24.2</td>
<td></td>
</tr>
<tr>
<td>REALM with random span masks</td>
<td>35.3</td>
<td>26.1</td>
<td></td>
</tr>
<tr>
<td>30× stale MIPS</td>
<td>28.7</td>
<td>15.1</td>
<td></td>
</tr>
</tbody>
</table>

4.4. Main results

Table 1 shows the accuracy of different approaches on the three Open-QA datasets. REALM outperform all previous approaches by a significant margin. Table 1 also shows the number of parameters for each model.

As reported in the concurrent work of Roberts et al. (2020), the generative Open-QA systems based on T5 are surprisingly powerful, with the largest T5-11B model outperforming the previous best Open-QA system. Increasing the size of T5 yields consistent improvement, but comes at significant computational cost (from Base to 11B, the model is 50 times larger, and gains roughly 7 points in accuracy). In contrast, REALM outperforms the largest T5-11B model while being 30 times smaller. It is also important to note that T5 accesses additional reading comprehension data from SQuAD during its pre-training (100,000+ examples). Access to such data could also benefit REALM, but was not used in our experiments.

Among all systems, the most direct comparison with REALM is ORQA (Lee et al., 2019) where the fine-tuning setup and training data are identical. We also include a version of ORQA with extended fine-tuning steps to match that of REALM fine-tuning, making all hyperparameters identical except for how the parameters were pre-trained. The improvement of REALM over ORQA is purely due to better pre-training. The results also indicate that our method of pre-training can be applied both on (1) the single-corpus setting ($X$ = Wikipedia, $Z$ = Wikipedia), or (2) the separate-corpus setting ($X$ = CC-News, $Z$ = Wikipedia).

Compared to other retrieval-based systems (Asai et al., 2019; Min et al., 2019a; b) which often retrieve from 20 to 80 documents, our system gets the overall best performance while only retrieving 5 documents.

4.5. Analysis

In Table 2 we present results for NaturalQuestions-Open after ablating critical components of REALM. In addition to the end-to-end results, we also report how often the gold answer appears in the top-5 retrievals before applying any
Table 3. An example where REALM utilizes retrieved documents to better predict masked tokens. It assigns much higher probability (0.129) to the correct term, “Fermat”, compared to BERT. (Note that the blank corresponds to 3 BERT wordpieces.)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) BERT</td>
<td>$p(y = \text{“Fermat”}</td>
<td>x) = 1.1 \times 10^{-14}$</td>
</tr>
<tr>
<td>(b) REALM</td>
<td>$p(y = \text{“Fermat”}</td>
<td>x, z) = 1.0$</td>
</tr>
<tr>
<td>(c) REALM</td>
<td>$p(y = \text{“Fermat”}</td>
<td>x) = 0.129$</td>
</tr>
</tbody>
</table>

5. Discussion and Related Work

We previously discussed related methods for Open-QA. Here we present several alternate ways of viewing REALM that connect it to a broader set of ideas beyond Open-QA:

Language modeling with corpus as context Language representation models have been incorporating contexts of increasingly large scope when making predictions. Examples of this progression include models that condition on surrounding words (Mikolov et al., 2013a;b), sentences (Kiros et al., 2015; Peters et al., 2018), and paragraphs (Radford et al., 2018; Devlin et al., 2018). We can view REALM as a generalization of the above work to the next level of scope: the entire text corpus.

Retrieve-and-edit with learned retrieval In order to better explain the variance in the input text and enable controllable generation, Guu et al. (2018) proposed a language model with the retrieve-and-edit framework (Hashimoto et al., 2018) that conditions on text with high lexical overlap. REALM has a similar approach, except that the model learns for itself which texts are most useful for reducing perplexity. By jointly learning the retriever, REALM has the capacity to depend on information beyond lexical overlap.

Scalable grounded neural memory The document index can be viewed as a memory where the keys are the document embeddings. From this view, our work share motivations with works such as product key memory (Lample et al., 2019), which enables sub-linear memory access in a memory network (Weston et al., 2014; Graves et al., 2014; Sukhbaatar et al., 2015), allowing these scalable memory layers to be integrated into large language models. One main difference is that our memories are grounded—each memory is associated with a document rather than unnamed value vectors. This level of interpretability is crucial for applications like Open-QA, where users would require provenance for a predicted answer to be trustworthy.

Unsupervised Corpus Alignment In sequence-to-sequence models with attention (Bahdanau et al., 2014), text is generated with latent selection of relevant tokens. This results in a set of model-centric unsupervised...
alignments between target and source tokens. Analogously, REALM also generates text with latent selection of relevant documents. A by-product of our method is that we offer a set of model-centric unsupervised alignments between text in the pre-training corpus $X$ and knowledge corpus $Z$.

6. Future Work

The work presented here is the minimal instantiation of a family of REALM-like approaches where a representation is pre-trained to perform reasoning over a large corpus of knowledge on-the-fly during inference. We are particularly optimistic about generalizations of this work to (1) structured knowledge, which would result in a generalization of Peters et al. (2019) where we would also learn the decision of which entities are informative, (2) the multi-lingual setting, e.g., retrieving knowledge in a high-resource language to better represent text in a low-resource language, and (3) the multi-modal setting, e.g., retrieving images or videos that can provide knowledge rarely observed in text.

References


