REALM: Retrieval-Augmented Language Model Pre-Training

Kelvin Guu [*1] **Kenton Lee** [*1] **Zora Tung** [1] **Panupong Pasupat** [1] **Ming-Wei Chang** [1]

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`").

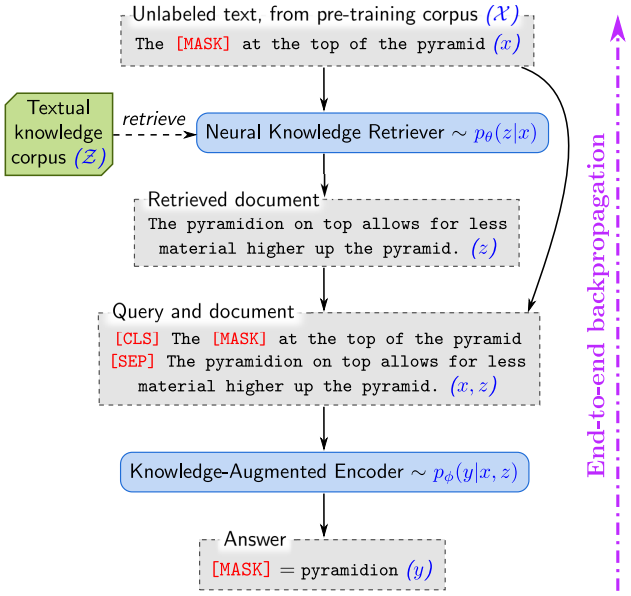

Figure 1. REALM augments language model pre-training with a **neural knowledge retriever** that retrieves knowledge from a **textual knowledge corpus**, $\mathcal{Z}$ (e.g., all of Wikipedia). Signal from the language modeling objective backpropagates all the way through the retriever, which must consider millions of documents in $\mathcal{Z}$—a significant computational challenge that we address.

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 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

[*]Equal contribution [1]Google Research. Correspondence to: Kelvin Guu <kguu@google.com>, Kenton Lee <kentonl@google.com>, Zora Tung <gatoatigrado@google.com>, Panupong Pasupat <ppasupat@google.com>, Ming-Wei Chang <mingweichang@google.com>.

decide what knowledge to retrieve and use during inference. Before making each prediction, the language model uses the retriever to retrieve documents[1] 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 model*[2] (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 = "`The [MASK] is the currency [MASK] the UK`"; y = ("`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 such as SQuAD (Rajpurkar et al., 2016; 2018). While RC mod-

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

[2] Strictly 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.

els comprehend a single document, Open-QA models must retain knowledge from millions of documents, since a question could be about any of them.

We focus on Open-QA systems that utilize a *textual knowledge corpus* $\mathcal{Z}$ 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 $\mathcal{Z}$, 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 discuss 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 $\mathcal{Z}$. 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

$$p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid z, x)\, p(z \mid x). \tag{1}$$

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:

$$p(z \mid x) = \frac{\exp f(x, z)}{\sum_{z'} \exp f(x, z')},$$
$$f(x, z) = \texttt{Embed}_{\texttt{input}}(x)^\top \texttt{Embed}_{\texttt{doc}}(z),$$

where $\texttt{Embed}_{\texttt{input}}$ and $\texttt{Embed}_{\texttt{doc}}$ 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 `[SEP]` tokens, prefixing a `[CLS]` token, and appending a final `[SEP]` token.

$$\text{query\_input}\quad \texttt{join}_{\texttt{BERT}}(x) = \texttt{[CLS]}\, x\, \texttt{[SEP]}$$
$$\text{doc}\quad \texttt{join}_{\texttt{BERT}}(x_1, x_2) = \texttt{[CLS]}\, x_1\, \texttt{[SEP]}\, x_2\, \texttt{[SEP]}$$

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

$$\text{query\_input}\quad \texttt{Embed}_{\texttt{input}}(x) = \mathbf{W}_{\texttt{input}}\texttt{BERT}_{\texttt{CLS}}(\texttt{join}_{\texttt{BERT}}(x))$$
$$\text{doc}\quad \texttt{Embed}_{\texttt{doc}}(z) = \mathbf{W}_{\texttt{doc}}\texttt{BERT}_{\texttt{CLS}}(\texttt{join}_{\texttt{BERT}}(z_{\text{title}}, z_{\text{body}}))$$

where z_{title} is the document's title and z_{body} is its body. We let θ 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 `[MASK]` token in x. To do so, we use the same masked

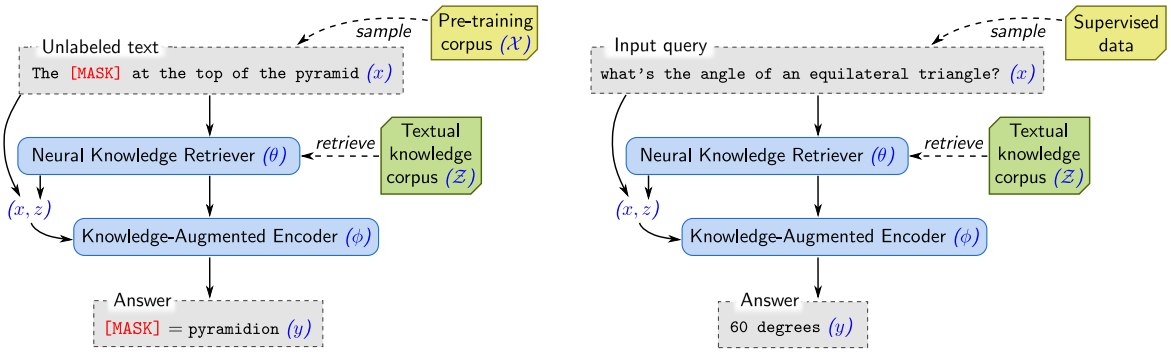

Figure 2. The overall framework of REALM. **Left:** *Unsupervised pre-training.* The knowledge retriever and knowledge-augmented encoder are jointly pre-trained on the unsupervised language modeling task. **Right:** *Supervised fine-tuning.* After the parameters of the retriever (θ) and encoder (ϕ) have been pre-trained, they are then fine-tuned on a task of primary interest, using supervised examples.

language modeling (MLM) loss as in Devlin et al. (2018):

$$p(y \mid z, x) = \prod_{j=1}^{J_x} p(y_j \mid z, x)$$

$$p(y_j \mid z, x) \propto \exp\left(w_j^\top \texttt{BERT}_{\texttt{MASK}(j)}(\texttt{join}_{\texttt{BERT}}(x, z_{\text{body}}))\right)$$

where $\texttt{BERT}_{\texttt{MASK}(j)}$ denotes the Transformer output vector corresponding to the j^{th} masked token, J_x 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 $p(y \mid z, x)$ as:

$$p(y \mid z, x) \propto \sum_{s \in S(z,y)} \exp\left(\texttt{MLP}\left(\left[h_{\texttt{START(s)}}; h_{\texttt{END(s)}}\right]\right)\right)$$

$$h_{\texttt{START(s)}} = \texttt{BERT}_{\texttt{START(s)}}(\texttt{join}_{\texttt{BERT}}(x, z_{\text{body}})),$$

$$h_{\texttt{END(s)}} = \texttt{BERT}_{\texttt{END(s)}}(\texttt{join}_{\texttt{BERT}}(x, z_{\text{body}})),$$

where $\texttt{BERT}_{\texttt{START(s)}}$ and $\texttt{BERT}_{\texttt{END(s)}}$ 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 ϕ 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 $\log p(y \mid x)$ of the correct output y. Since both the knowledge retriever and knowledge-augmented encoder are differentiable neural networks, we can compute the gradient of $\log p(y \mid x)$ (defined in Equation 1) with respect to the model parameters θ and ϕ, and optimize using stochastic gradient descent.

The key computational challenge is that the marginal probability $p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid x, z)\, p(z \mid x)$ involves a summation over all documents z in the knowledge corpus $\mathcal{Z}$. We approximate this by instead summing over the top k documents with highest probability under $p(z \mid x)$—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 $p(z \mid x)$ is the same as under the relevance score $f(x, z) = \texttt{Embed}_{\texttt{input}}(x)^\top \texttt{Embed}_{\texttt{doc}}(z)$, 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 $\texttt{Embed}_{\texttt{doc}}(z)$ for every $z \in \mathcal{Z}$ and construct an efficient search index over these embeddings. However, this data structure will no longer be consistent with $p(z \mid x)$ if the parameters θ of $\texttt{Embed}_{\texttt{doc}}$ are later updated. Hence, the search index goes "stale" after every gradient update on θ.

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 $p(z \mid x)$ and its gradient, using the fresh θ, 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

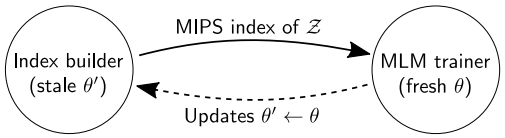

Figure 3. REALM pre-training with asynchronous MIPS refreshes.

below, the trainer sends the index builder a snapshot of its parameters, θ'. The trainer then continues to train while the index builder uses θ' to construct a new index in the background. As soon as the index builder is done, it sends the new index back to the trainer, and the process repeats.

While asynchronous refreshes can be used for both pre-training and fine-tuning, in our experiments we only use it for pre-training. For fine-tuning, we just build the MIPS index once (using the pre-trained θ) for simplicity and do not update $\texttt{Embed}_{\texttt{doc}}$.[3] Note that we still fine-tune $\texttt{Embed}_{\texttt{input}}$, so the retrieval function is still updated from the query side.

What does the retriever learn? Since the knowledge retrieval of REALM is latent, it is not obvious how the training objective encourages meaningful retrievals. Here, we show how it rewards retrievals that improve prediction accuracy.

For a given query x and document z, recall that $f(x, z)$ is the "relevance score" that the knowledge retriever assigns to document z. We can see how a single step of gradient descent during REALM pre-training alters this score by analyzing the gradient with respect to the parameters of the knowledge retriever, θ:

$$\nabla \log p(y \mid x) = \sum_{z \in \mathcal{Z}} r(z) \nabla f(x, z)$$

$$r(z) = \left[\frac{p(y \mid z, x)}{p(y \mid x)} - 1 \right] p(z \mid x).$$

For each document z, the gradient encourages the retriever to change the score $f(x, z)$ by $r(z)$ — increasing if $r(z)$ is positive, and decreasing if negative. The multiplier $r(z)$ is positive if and only if $p(y \mid z, x) > p(y \mid x)$. The term $p(y \mid z, x)$ is the probability of predicting the correct output y when using document z. The term $p(y \mid x)$ is the expected value of $p(y \mid x, z)$ when randomly sampling a document from $p(z \mid x)$. Hence, document z receives a positive update whenever it performs better than expected.

[3] This works because pre-training already yields a good $\texttt{Embed}_{\texttt{doc}}$ function. However, it is possible that refreshing the index would further improve performance.

3.4. Injecting inductive biases into pre-training

In the process of developing REALM, we discovered several additional strategies that further guide the model towards meaningful retrievals, described below.

Salient span masking During REALM pre-training, we want to focus on examples x that require world knowledge to predict the masked tokens. As explained in Section 2, some MLM spans only require local context. To focus on problems that require world knowledge, we mask *salient spans* such as "`United Kingdom`" or "`July 1969`". We use a BERT-based tagger trained on CoNLL-2003 data (Sang & De Meulder, 2003) to identify named entities, and a regular expression to identify dates. We select and mask one of these salient spans within a sentence for the masked language modeling task. We show that this significantly outperforms other masking strategies in Section 4.5.

Null document Even with salient span masking, not all masked tokens require world knowledge to predict. We model this by adding an empty *null document* $\varnothing$ to the top k retrieved documents, allowing appropriate credit to be assigned to a consistent sink when no retrieval is necessary.

Prohibiting trivial retrievals If the pre-training corpus $\mathcal{X}$ and the knowledge corpus $\mathcal{Z}$ are the same, there exists a trivial retrieval candidate z that is *too* informative: if the masked sentence x comes from document z, the knowledge augmented encoder can trivially predict y by looking at the unmasked version of x in z. This results in a large positive gradient for $p(z \mid x)$. If this occurs too often, the knowledge retriever ends up learning to look for exact string matches between x and z, which does not capture other forms of relevance. For this reason, we exclude this trivial candidate during pre-training.

Initialization At the beginning of training, if the retriever does not have good embeddings for $\texttt{Embed}_{\texttt{input}}(x)$ and $\texttt{Embed}_{\texttt{doc}}(z)$, the retrieved documents z will likely be unrelated to x. This causes the knowledge augmented encoder to learn to ignore the retrieved documents. Once this occurs, the knowledge retriever does not receive a meaningful gradient and cannot improve, creating a vicious cycle. To avoid this cold-start problem, we warm-start $\texttt{Embed}_{\texttt{input}}$ and $\texttt{Embed}_{\texttt{doc}}$ using a simple training objective known as the Inverse Cloze Task (ICT) where, given a sentence, the model is trained to retrieve the document where that sentence came from. We defer to Lee et al. (2019) for details. For the knowledge-augmented encoder, we warm-start it with BERT pre-training—specifically, the uncased BERT-base model (12 layers, 768 hidden units, 12 attention heads).

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).[4] We compare against the Base, Large, and even larger 11-billion parameter model to measure the effect of model size.

4.3. Implementation Details

Fine-tuning We reuse all hyperparameters from Lee et al. (2019), to enable direct comparison. Our 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 candidates. During fine-tuning inference, we consider the top-5 candidates, and the

[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 paretheses 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.

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

Table 2. Ablation experiments on NQ's development set.

Ablation	Exact Match	Zero-shot Retrieval Recall@5
REALM	38.2	38.5
REALM retriever+Baseline encoder	37.4	38.5
Baseline retriever+REALM encoder	35.3	13.9
Baseline (ORQA)	31.3	13.9
REALM with random uniform masks	32.3	24.2
REALM with random span masks	35.3	26.1
30× stale MIPS	28.7	15.1

entire model can be run on a single machine with a 12GB GPU.

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 $\varnothing$.

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).

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 5 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, hyperparameters and training data are identical. The improvement of REALM over ORQA is purely due to better pre-training methods. The results also indicate that our method of pre-training can be applied both on (1) the single-corpus setting ($\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia), or (2) the separate-corpus setting ($\mathcal{X}$ = CC-News, $\mathcal{Z}$ = 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 fine-tuning. The latter metric more significantly isolates the contribution of improving the retriever during pre-training.

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.)

x:	An equilateral triangle is easily constructed using a straightedge and compass, because 3 is a ___ prime.		
(a) BERT	$p(y = \text{"Fermat"} \mid x)$	$= 1.1 \times 10^{-14}$	(No retrieval.)
(b) REALM	$p(y = \text{"Fermat"} \mid x, z)$	$= 1.0$	(Conditional probability with document z ="257 is ... a Fermat prime. Thus a regular polygon with 257 sides is constructible with compass ...")
(c) REALM	$p(y = \text{"Fermat"} \mid x)$	$= 0.129$	(Marginal probability, marginalizing over top 8 retrieved documents.)

Encoder or Retriever We first aim to determine whether REALM pre-training improves the retriever or the encoder, or both. To do so, we can reset the parameters of either the retriever or the encoder to their baseline state before REALM pre-training, and feed that into fine-tuning. Resetting both the retriever and encoder reduces the system to our main baseline, ORQA. We find that both the encoder and retriever benefit from REALM training separately, but the best result requires both components acting in unison.

Masking scheme We compare our salient span masking scheme (Section 3.4) with (1) random token masking introduced in BERT (Devlin et al., 2018) and (2) random span masking proposed by SpanBERT (Joshi et al., 2019). While such salient span masking has not been shown to be impactful in previous work with standard BERT training (Joshi et al., 2019), it is crucial for REALM. Intuitively, the latent variable learning relies heavily on the utility of retrieval and is therefore more sensitive to a consistent learning signal.

MIPS index refresh rate During pre-training, we run a parallel process to re-embed corpus documents and rebuild the MIPS index. This results in one index refresh per approximately 500 training steps. To demonstrate the importance of frequent index refreshes, we compare against using a slower refresh rate. The results in Table 2 suggests that a stale index can hurt model training, and further reducing this staleness could offer better optimization.

Examples of retrieved documents Table 3 shows an example of the REALM masked language model prediction. In this example, "Fermat" is the correct word, and REALM (row (c)) gives the word a much high probability compared to the BERT model (row (a)). Since REALM manages to retrieve some documents with a related fact (row (b)), the marginalized probability of the correct answer dramatically increases. This shows that REALM is able to retrieve document to fill in the masked word even though it is trained with unsupervised text only.

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 $\mathcal{X}$ and knowledge corpus $\mathcal{Z}$.

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

Asai, A., Hashimoto, K., Hajishirzi, H., Socher, R., and Xiong, C. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*, 2019.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, 2013.

Brill, E., Dumais, S., and Banko, M. An analysis of the askmsr question-answering system. In *Empirical Methods in Natural Language Processing*, 2002.

Chen, D., Fisch, A., Weston, J., and Bordes, A. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 1870–1879, 2017.

Clark, C. and Gardner, M. Simple and effective multi-paragraph reading comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

Dai, A. M. and Le, Q. V. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pp. 3079–3087, 2015.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *ArXiv*, abs/1410.5401, 2014.

Guu, K., Hashimoto, T. B., Oren, Y., and Liang, P. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450, 2018.

Hashimoto, T. B., Guu, K., Oren, Y., and Liang, P. S. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pp. 10052–10062, 2018.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Generalization through memorization: Nearest neighbor language models. *ArXiv*, abs/1911.00172, 2019.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. Skip-thought vectors. In *Advances in neural information processing systems*, pp. 3294–3302, 2015.

Kwiatkowski, T., Palomaki, J., Rhinehart, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 2019.

Lample, G., Sablayrolles, A., Ranzato, M., Denoyer, L., and Jégou, H. Large memory layers with product keys. In *Advances in Neural Information Processing Systems*, pp. 8546–8557, 2019.

Lee, K., Salant, S., Kwiatkowski, T., Parikh, A., Das, D., and Berant, J. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016.

Lee, K., Chang, M.-W., and Toutanova, K. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the Conference of Association for Computational Linguistics*, 2019.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv*, abs/1910.13461, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013b.

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

Min, S., Chen, D., Hajishirzi, H., and Zettlemoyer, L. A discrete hard em approach for weakly supervised question answering. *arXiv preprint arXiv:1909.04849*, 2019a.

Min, S., Chen, D., Zettlemoyer, L., and Hajishirzi, H. Knowledge guided text retrieval and reading for open domain question answering. *arXiv preprint arXiv:1911.03868*, 2019b.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

Peters, M. E., Neumann, M., IV, R. L. L., Schwartz, R., Joshi, V., Singh, S., and Smith, N. A. Knowledge enhanced contextual word representations, 2019.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding with unsupervised learning. Technical report, OpenAI, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

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

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:TBD*, 2020.

Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Sang, E. T. K. and De Meulder, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*, 2016.

Shen, F., Liu, W., Zhang, S., Yang, Y., and Tao Shen, H. Learning binary codes for maximum inner product search. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4148–4156, 2015.

Shrivastava, A. and Li, P. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pp. 2321–2329, 2014.

Sukhbaatar, S., Weston, J., Fergus, R., et al. End-to-end memory networks. In *Advances in neural information processing systems*, 2015.

Weston, J., Chopra, S., and Bordes, A. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

A. Derivation of the gradient with respect to the knowledge retriever

We compute the gradient of the REALM pre-training objective (a log-likelihood) with respect to the parameters of the knowledge retriever, θ:

$$\begin{aligned}\nabla \log p(y \mid x) &= p(y \mid x)^{-1} \nabla p(y \mid x) \\ &= p(y \mid x)^{-1} \sum_z p(y \mid z, x) \nabla p(z \mid x) \\ &= p(y \mid x)^{-1} \sum_z p(y \mid z, x) p(z \mid x) \nabla \log p(z \mid x) \\ &= \sum_z p(z \mid y, x) \nabla \log p(z \mid x),\end{aligned}$$

where the last line follows from applying conditional Bayes' rule. We can then expand $\nabla \log p\left(z \mid x\right)$ as:

$$\begin{aligned}\nabla \log p(z \mid x) &= \nabla \log \frac{\exp f(x, z)}{\sum_{z'} \exp f(x, z')} \\ &= \nabla \left[f(x, z) - \log \sum_{z'} \exp f(x, z') \right] \\ &= \nabla f(x, z) - \sum_{z'} p(z' \mid x) \nabla f(x, z')\end{aligned}$$

Plugging this back into the first set of equations yields:

$$\begin{aligned}\nabla \log p\left(y \mid x\right) &= \sum_z p\left(z \mid y, x\right) \left[\nabla f(x, z) - \sum_{z'} p\left(z' \mid x\right) \nabla f(x, z') \right] \\ &= \sum_z p\left(z \mid y, x\right) \nabla f(x, z) - \sum_{z'} p\left(z' \mid x\right) \nabla f(x, z') \\ &= \sum_z \left[p\left(z \mid y, x\right) - p\left(z \mid x\right) \right] \nabla f(x, z) \\ &= \sum_z \left[\frac{p\left(y \mid z, x\right) p\left(z \mid x\right)}{p\left(y \mid x\right)} - p\left(z \mid x\right) \right] \nabla f(x, z) \\ &= \sum_z \left[\frac{p\left(y \mid z, x\right)}{p\left(y \mid x\right)} - 1 \right] p\left(z \mid x\right) \nabla f(x, z).\end{aligned}$$

In the second line, we used the fact that the overall expression is an expectation with respect to $p\left(z \mid y, x\right)$, and the terms which depend on z' but not z can be moved out of that expectation.

B. Connection between REALM and supervised learning

From the equations in Appendix A, we saw that

$$\nabla \log p\left(y \mid x\right) = \sum_z \left[p\left(z \mid y, x\right) - p\left(z \mid x\right) \right] \nabla f(x, z).$$

Suppose that there exists one document z^* which causes the model to achieve perfect prediction accuracy (i.e., $p\left(y \mid z^*, x\right) = 1$), while all other documents z' result in zero accuracy (i.e., $p\left(y \mid z', x\right) = 0$). Under this setting, $p\left(z^* \mid y, x\right) = 1$ (provided that $p\left(z^* \mid x\right)$ is non-zero), which causes the gradient to become

$$\begin{aligned}\nabla \log p\left(y \mid x\right) &= \nabla f\left(x, z^*\right) - \sum_z p\left(z \mid x\right) \nabla f(x, z) \\ &= \nabla \log p\left(z^* \mid x\right).\end{aligned}$$

From this, we see that gradient descent on the REALM objective is equivalent to gradient descent on $\log p\left(z^* \mid x\right)$. This is none other than the typical maximum likelihood training objective used in supervised learning, where z^* is the "gold" document.

C. Adapting to new knowledge

An explicit retrieval system allows us to adapt to new world knowledge simply by modifying the corpus documents. To demonstrate this ability, we replace the knowledge corpus with a more recent version of Wikipedia corpus after pre-training is done. When the input query is about a fact where the two corpora disagree, REALM can change the prediction to reflect the updated information, as exemplified in Table 4. However, even with an explicit retrieval mechanism, the knowledge-augmented encoder will still end up remembering some world knowledge, making the prediction of some input sentences not updated with the new corpus. (For instance, the model predicts "`Thatcher`" for "`___ is the prime minister of United Kingdom.`" on both corpora, perhaps due to the frequent mention of her name in Wikipedia articles.)

D. Retrieval Utility

The null document $\varnothing$ described in Section 3.4 provides a way to measure the importance of a retrieved document z: we define the *retrieval utility* (RU) of z for the masked input x as the difference between the log-likelihood of the knowledge-augmented encoder when conditioning on z versus on $\varnothing$:

$$\text{RU}(z \mid x) = \log p(y \mid z, x) - \log p(y \mid \varnothing, x). \quad (2)$$

A negative RU shows that z is less useful for predicting y than the null document. This could mean that z is irrelevant to x, but could also mean that the masked tokens in x do not require world knowledge to predict, or that the world knowledge is sufficiently commonplace it has been baked into the model's parameters. In practice, we find that RU increases steadily over the course of pre-training, and is more predictive of good performance on the downstream task of Open-QA than even the overall log-likelihood. An example of how RU behaves over time and across different settings is in Figure 4.

x:	"Jennifer ___ formed the production company Excellent Cadaver."
BERT	also (0.13), then (0.08), later (0.05), ...
REALM ($\mathcal{Z}$ =20 Dec 2018 corpus)	smith (0.01), brown (0.01), jones (0.01)
REALM ($\mathcal{Z}$ =20 Jan 2020 corpus)	**lawrence** (0.13), brown (0.01), smith (0.01), ...

Table 4. An example where REALM adapts to the updated knowledge corpus. The Wikipedia page "Excellent Cadaver" was added in 2019, so the model was not about to recover the word when the knowledge corpus is outdated (2018). Interestingly, the same REALM model pre-trained on the 2018 corpus is able to retrieve the document in the updated corpus (2020) and generate the correct token, "Lawrence".

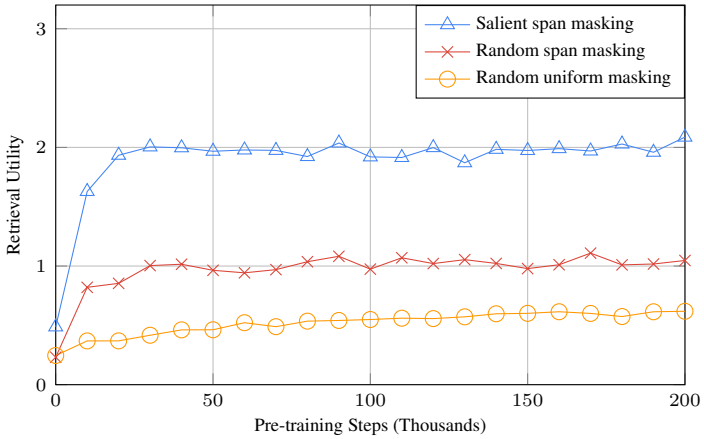

Figure 4. The Retrieval Utility (RU, described in Eq. 2) vs the number of pre-training steps. RU roughly estimates the "usefulness" of retrieval. RU is impacted by the choice of masking and the number of pre-training steps.