REALM: Retrieval-Augmented Language Model Pre-Training

2024.09.30

Yonsei Univ.
Multimodal AI Lab

Agenda

1. Introduction
2. Overview
3. Background
4. Model
5. Result

Introduction

- Recent LM pre-training method can learn world knowledge *implicitly*. (e.g. BERT, RoBERTa)
  - It makes difficult to determine what knowledge is stored in the networks and where.
  - Furthermore, storage space is limited to capture more knowledge.

- So we introduce **REALM pre-training**, with textual knowledge retriever
  - Before making each prediction, LM uses the retriever to retrieve docs from a large corpus. (aka. retrieve-then-predict)
  - Learning this model end-to-end (backpropagating through a retrieval step with performance-based signal)
    - "The __ at the top of the pyramid" → "The pyramidon on top allows for less material higher up the pyramid"

Overview

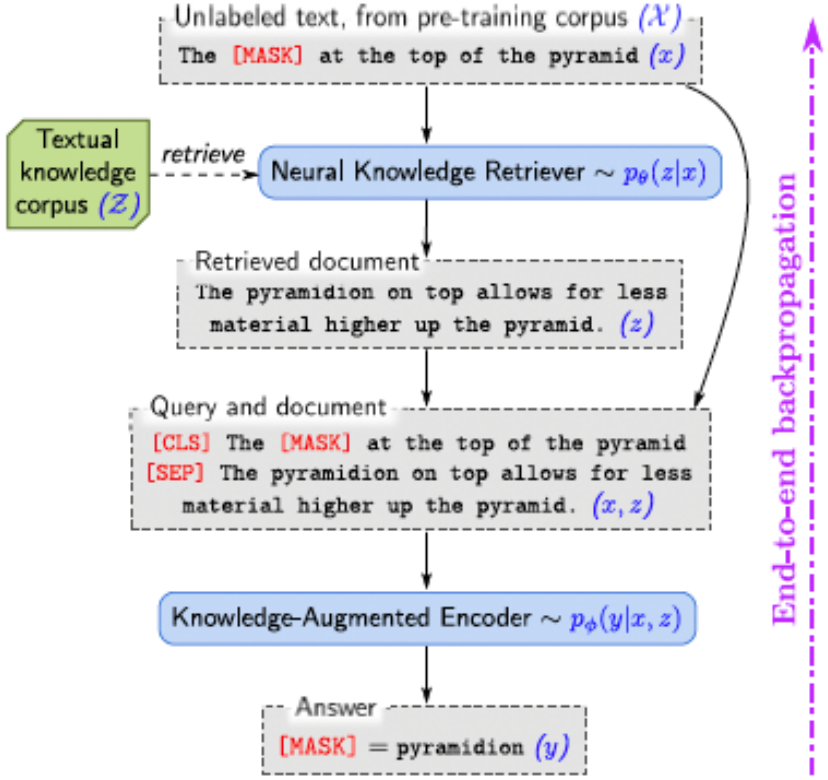

Figure 1. REALM augments language model pre-training with a **neural knowledge retriever** that retrieves knowledge from a **textual knowledge corpus**, $\mathcal{Z}$ (e.g., all of Wikipedia). Signal from the language modeling objective backpropagates all the way through the retriever, which must consider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

- Main Contribution
  - Jointing Retriever & Reader for end-to-end model
    - Prior works demonstrated benefit of using discrete retrieval steps, but they did not use for pre-training and used only non-learned retrievers.
  - Retrieve-then-predict
    - Neural Knowledge Retriever : Query → relevant document
    - Knowledge-Augmented Encoder : retrieved doc → answer
  - Pre-training & fine-tuning
    - MLM in Pre-training
    - Open-domain QA in fine-tuning

Background

- Language Model pre-training
  - Goal of pre-training is to learn useful representations of language from unlabeled text.
  - We focus on the *masked language model* (MLM) which is popularized by BERT
    - e.g. x = “The [MASK] is the currency [MASK] the UK” → y = (“pound”, “of”)

- Open-domain QA
  - e.g. “What is the currency of the UK” → “pound”
  - “Open” refers to the fact that the model does not receive a pre-identified document that is know to contain the answer, unlike traditional *reading comprehension (RC)* tasks such as SQuAD.

Model : generative process

- REALM decomposes $p(z|x)$ into two steps (retrieve & predict)
  - For given input x, we first retrieve helpful document z from a corpus $\mathcal{Z}$: $p(z|x)$
  - Then, condition both x, z to generate the answer y : $p(y|\, z, x)$

  - To obtain the overall likelihood of generating y, we treat z as a latent variable and marginalize over all possible documents.

$$p(y|x) = \sum_{z \in \mathcal{Z}} p(y|z,x)p(z|x)$$

Model Architecture

- **Knowledge Retriever** : $p(z|x)$
  - Retriever is defined using a dense inner product model & softmax
  - Let f : relevant score, $Embed$: embedding function that maps x and z to d-dimensional vectors.

$$p(z|x) = \frac{e^{f(x,z)}}{\sum_{z'} e^{f(x,z')}}$$

$$f(x,z) = \text{Embed}_{input}(x) \cdot \text{Embed}_{doc}(z)$$

  - Implementing Embedding function with BERT-style Transformer : Spans one vector for each token
    - Joining spans with [SEP] tokens, prefixing [CLS] token
    - project the vector of [CLS] token with projection matrix W_{input}, W_{doc}

$$join_{\text{BERT}}(x) = [\text{CLS}]x[\text{SEP}]$$

$$join_{\text{BERT}}(x_1, x_2) = [\text{CLS}]x_1[\text{SEP}]x_2[\text{SEP}]$$

$$\longrightarrow$$

$$\text{Embed}_{input}(x) = \boldsymbol{W}_{input}\ \text{BERT}_{\text{CLS}}(join_{\text{BERT}}(x))$$

$$\text{Embed}_{doc}(z) = \boldsymbol{W}_{doc}\ \text{BERT}_{\text{CLS}}(join_{\text{BERT}}(z_{title}, z_{body}))$$

Model Architecture

- **Knowledge Augmented Encoder** : $p(y|\, z, x)$
  - Joining x and z into a single sequence and feed it to Transformer. (For Cross-attention)
  - Architecture for pre-training and fine-tuning differs slightly

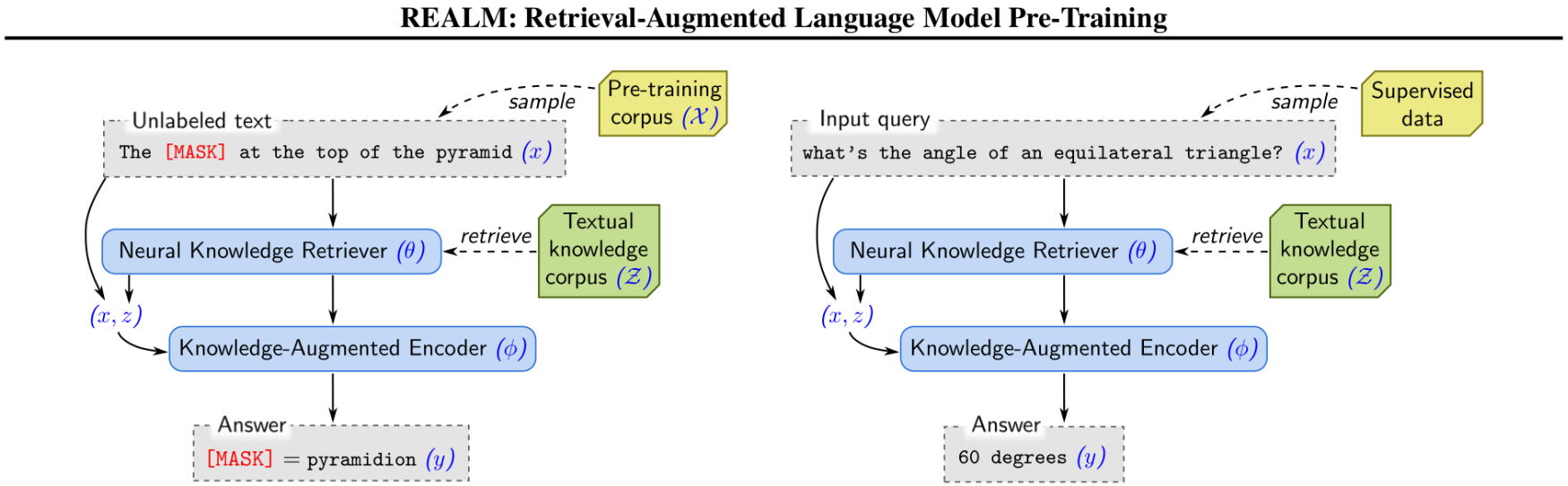

Figure 2. The overall framework of REALM. **Left:** *Unsupervised pre-training.* The knowledge retriever and knowledge-augmented encoder are jointly pre-trained on the unsupervised language modeling task. **Right:** *Supervised fine-tuning.* After the parameters of the retriever (θ) and encoder (ϕ) have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

Model Architecture

- **Knowledge Augmented Encoder : Pre-training**
  - Using Masked language modeling (MLM) loss
  - Let J_x : total number of [MASK] tokens in x, $BERT_{MASK(j)}$: output token of j^{th} masked token

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

$$p(y_j|z,x) \propto exp(w_j \cdot \mathrm{BERT}_{\mathrm{MASK(j)}}(join_{\mathrm{BERT}}(x, z_{body})))$$

Model Architecture

- **Knowledge Augmented Encoder : fine-tuning**
  - Assume that the answer token y can be found as a contiguous sequence of tokens in some document z
  - Let $\boldsymbol{S(z,y)}$: set of spans matching y in z, (spans can found in several place)

$$p(y|z,x) \propto \sum_{s \in S(z,y)} exp(\text{MLP}(h_{START(s)}; h_{END(s)}))$$

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

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

Model : Training

- For both pre-training & fine-tuning, we train by maximizing the log-likelihood **log p(y | x)**
- But if we calculate the gradient for the equation below, it causes high computational power

$$p(y|x) = \sum_{z \in \mathcal{Z}} p(y|z,x)p(z|x)$$

  - So we approximate this calculation by using only top-k documents
  - Reasonable if most documents have near zero probability
- Even with this approximation, we still need an efficient way to find the top-k documents.

MIPS (Maximum Inner Product Search) Algorithm

Model Training

- **Implementing asynchronous MIPS refreshes**
  - To employ MIPS, we must pre-compute $Embed_{doc}(z)$ for every $z \in \mathcal{Z}$ and construct an efficient search index over these embeddings. (e.g. LSH, ANN, ScaNN)
  - But these data (index) will be **"stale"** during the training process
  - Our Solution is to **"refresh"** the index by **asynchronously** re-embedding and re-indexing all docs every several training steps.

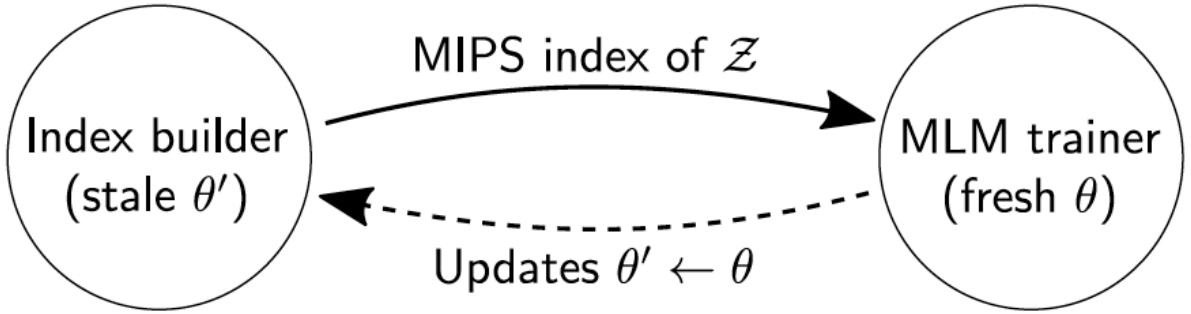

Figure 3. REALM pre-training with asynchronous MIPS refreshes.

Model Training

- **What does the retriever learn?** (IOW, How can retriever learn by end-to-end learning?)

$$\nabla p(y|x) \quad \text{---------- Mathematical something.. ----------} \rightarrow \quad \nabla \log p(y|x) = \sum_{z \in \mathcal{Z}} r(z)\ \nabla f(x,z)$$

$$r(z) = \left[\frac{f(y|z,x)}{f(y|x)} - 1 \right] f(z|x)$$

- If the probability for predicting y with condition x, z is larger than that of only with x, the gradient will be positive.
- That is, if z is helpful for predicting y, it will be rewarded.

Model : injecting inductive bias

- Several additional strategies that guide the model towards meaningful retrievals. (In pre-training)
  - **Salient span masking**
    - To focus on examples x that require world knowledge, we mask *salient spans* such as “United Kingdom” or “July 1969”
    - Using BERT-based tagger trained on CoNLL-2003 dataset & regular expression
  - **Null document**
    - Even with salient span masking, not all masked tokens require world knowledge to predict.
    - Adding null document to the top-k documents, it can help predicting tokens which does not need additional information.

Model : injecting inductive bias

- Several additional strategies that guide the model towards meaningful retrievals. (In pre-training)
  - **Prohibiting trivial retrievals**
    - If the pre-training corpus $\mathcal{X}$ and knowledge corpus $\mathcal{Z}$ are the same, there exists a trivial retrieval candidate z that is too informative
    - This can occur the retrievers learn to capture exact string match. Thus, we exclude this trivial candidate during pre-training
  - **Initialization**
    - At the beginning of training, if the retriever does not have good embeddings, the retrieved doc z will likely be unrelated to x. This can cause cold-start problem, learn to ignore the retrieved document.
    - By simply training embedding params by ICT, we can make useful initial params

Result

REALM: Retrieval-Augmented Language Model Pre-Training

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

Using Only top-5 passage & 30 times smaller model than T5(11b) model.

Result

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

Ablation	Exact Match	Zero-shot Retrieval Recall@5
REALM	38.2	38.5

- Ablation Study
  - Retriever + Encoder
  - Masking Method
  - Refreshing frequency

REALM: Retrieval-Augmented Language Model 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.)