Case Studies on State-of-the-Art Retrieval Based Language Models

Tianjian Li

Johns Hopkins University

June 8, 2022

1 REALM

REALM is proposed in **REALM: Retrieval-Augmented Language Model Pre-Training**[3], published at ICML 2020. It proposes a pipeline that incorporates relevant document retrieval to augment knowledge probing in large language models. It is an modification to the masked language modeling pretrain method proposed in BERT[1], as well as a finetune method used to query real world knowledge inside language models. The paper's major contributions are:

- Proposing a novel pretrain method by using document retrieval to allow language models attend to factual knowledge in documents.
- Solving the major problems that arises from measurement of document-sentence similarity, and giving out explanations of their methodology.

As we know, large language models stores factual knowledge within its parameters. We would like to explicitly retrieve the knowledge to perform Question-Answering(QA) or other tasks that require knowledge probing. REALM augments the **pre-train** objective with a learned knowledge retriever, so that during inference, the model can retrieve relevant documents from a large corpus.

1.1 Pipeline and Implementation Choices

Generally speaking, Language Models aims to optimize the conditional distribution of the next token given a context. $p(y|x)$. REALM first retrieve useful document from a knowledge corpus Z, which is measured by $p(z|x)$, and $z \in Z$. Then the model is pre-trained to generate the output y conditioned on z and x, that is $p(y|z, x)$.

The retriever is defined using a inner product model.

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

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

Here $\text{Embed}_{\text{input}}$ and $\text{Embed}_{\text{doc}}$ are embedding functions that maps an input sentence x and a document z to a vector $v \in R^d$, the relevance score is computed by the dot product of the embeddings, then all scores are softmaxed. The embedding is a linear projection of the BERT [CLS] token of the input and document.

After a document is retrieved, the model concatenates the input x and the document z and passes them into a transformer model. During pre-training, the authors uses the same objective of BERT[1], which predicts the word/token at [MASK] positions. During fine-tuning, assuming that the answer can be explicitly found in a document rather than being generated, we can define $p(y|z, x)$ as:

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

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

where $S(z, y)$ denotes the set of spans matching y in z. Honestly I am confused by the notations here but I think it means that given a set of spans, we use a BERT model to encode the starting token and ending token into vectors, concatenate them and then feed them into a feed forward network to compute the score of a span. The "score" of a particular set of spans(set of answers) is summed by the score of each single span.

To implement REALM, one of the biggest challenges are to compute the softmaxed document score within a huge amount of documents. So the author pre-computes the scores of the document and performs a Maximum Inner Product Search(MIDS). However, as the parameters of the document retriever(which is a BERT model) gets updated, the pre-computed vectors, and therefore the dot product scores becomes stale. The author solve(or attempts to solve) this problem by asynchronously updating the retriever(BERT encoder) parameters Once in a while, the model sends the update of the retriever parameters to the document vector pre-computing agency and let it update the stale vectors.

1.2 My Thoughts

This paper is the pioneering work of using retrieval to augment language model pretraining to let it perform better in knowledge probing. I believe that some of its engineering choices could be better(e.g. selecting the encoding of start and end of spans). The generalizability of this method is to be examined. Using retrieval only during fine-tuning to get answer from a large document or collection of document is of great potential.

2 DPR

Densed Passage Retrieval(DPR) is proposed in **Dense Passage Retrieval for Open-Domain Question Answering**[4], published at EMNLP 2020. This paper empirically finds that "in the context of open-domain question answering, a higher retrieval precision indeed translates to a higher end-to-end QA accuracy."[4]. Therefore training a high accuracy retriever is equivalent to training an end to end model that maximizes the likelihood of the ground-truth answer. The main contributions of this paper are:

- Proposing Dense Embedding(A Neural Retriever) can be a better solution for retrieval than TF-IDF or BM25.
- Proposing a more efficient neural retriever than ORQA[2] since the latter needs to compute all of conditional probabilities of contexts.
- Proposing in batch negative sampling to make training efficient, experimenting with various sampling strategies.
- Empirically studying the positive relationship between a better retriever and a higher QA accuracy.

The model is very simple: First use BERT[1] Encoders to get a d dimensional vector representation of the query and documents.[1] Use inner product between the query vector and the document vector to retrieve documents.

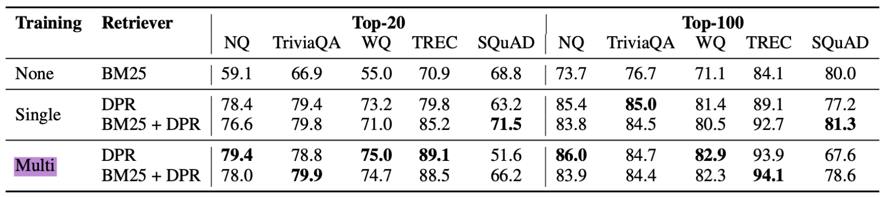
During Training, the model optimizes the negative log likelihood of the positive passage:

$$L = -\log \frac{e^{sim(q,p^+)}}{e^{sim(q,p^+)} + \sum e^{sim(q,p^-)}}$$

Additionally, the author proposes in-batch negatives: given a batch of B questions, each associated with a relevant passage, we have B^2 question-passage pairs. Positive sample is the relevant passage while all the other passages within the same batch are negative passages.

[1]The authors use [CLS] as the encoded vector, therefore d = 768

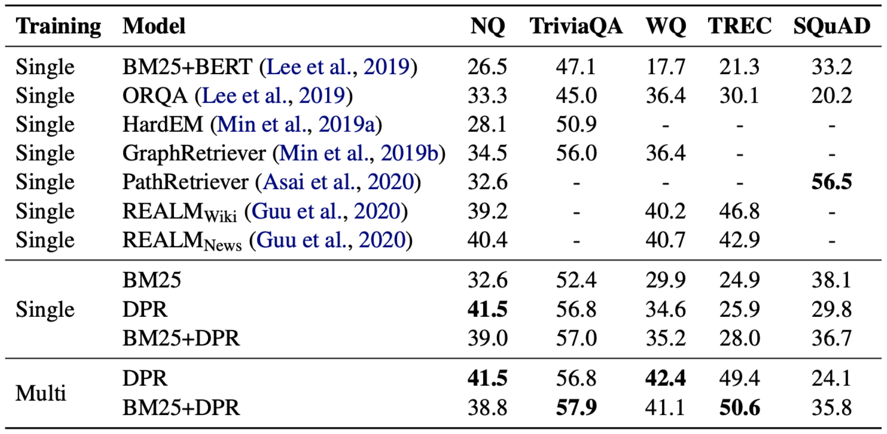
2.1 My Thoughts



Training	Retriever	Top-20					Top-100				
		NQ	TriviaQA	WQ	TREC	SQuAD	NQ	TriviaQA	WQ	TREC	SQuAD
None	BM25	59.1	66.9	55.0	70.9	68.8	73.7	76.7	71.1	84.1	80.0
Single	DPR	78.4	79.4	73.2	79.8	63.2	85.4	**85.0**	81.4	89.1	77.2
	BM25 + DPR	76.6	79.8	71.0	85.2	**71.5**	83.8	84.5	80.5	92.7	**81.3**
Multi	DPR	**79.4**	78.8	**75.0**	**89.1**	51.6	**86.0**	84.7	**82.9**	93.9	67.6
	BM25 + DPR	78.0	**79.9**	74.7	88.5	66.2	83.9	84.4	82.3	**94.1**	78.6

Figure 1: Retrieval Accuracy, copied from [4]

The Experiment results shows 1)DPR outperforms non neural based retriever(BM25) in retrieval accuracy. 2) Adding more retrieved passages increase retrieval accuracy. Here multi means training with multiple datasets.



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

Figure 2: EM scores on OpenQA benchmarks, copied from [4]

I am a little skeptical about the results of DPR after examining this table. First, DPR did not outperform REALM[3] under single dataset pretrain setting, actually the difference is quite high(about 10%). This suggests that pretraining a neural retriever on multiple datasets is very crucial to its performance. Secondly, it is not a fair comparison because each different model uses different ways to generate answer spans. REALM uses the concatenation of start and end token to represent spans while DPR uses the product of linear transformed representation of start token and end token to represent spans. The difference in their way of decoding makes it not a fair comparison.

3 RAG

RAG is proposed in **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**[6], published at NIPS 2020. This paper identifies the problem of "their(Large Language Model's) ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their(LLM's) performance lags behind task-specific architectures."[6]. The main contributions of this paper are:

1. This paper is the first work(to the best of my knowledge) to in-cooperate a sequence to sequence structure to autoregressively generate the answer span.
2. Providing comparisons to existing models(BM25, DPR[4], REALM[3]) on open domain QA, abstract QA and Question Generation.

	Retriever	Decoder	Training Objective	Remarks

GraphQA	Graph Based	MLP	Marginal Likelihood	Vertices = Passages
				Edges = BM25 Similarity

ORQA(ACL '19)	BERT[CLS]	BERT Concat	Inverse Cloze Task(ICT)	Softmax operation computationally expensive
REALM(ICML '20)	BERT[CLS]	BERT Concat	Masked Language Model	Document encoder updated asynchronously
DPR(EMNLP '20)	BERT[CLS]	BERT Multiply	Neg. log prob. of positive documents	Only the retriever is trained Question = Gold Data
RAG(NIPS '20)	BERT[CLS]	BART AR*	Marginal Likelihood	Document encoder not updated Seq2Seq autoregressive decoder
ProQA(EACL '21)	BERT[CLS]	BERT	Neg. log prob. of positive documents + answer scores	Questions = BART generations Trained end-to-end. Treats all retrieved docs as a huge doc.

Table 1: Comparison between different retrieval augmented LMs, *Autoregressive

Similarly to REALM and DPR, RAG also uses seperate BERT encoder for query and document, then computes the dot product of query embedding and document embedding for ranking the documents. Unlike REALM, which asynchronously updates the document encoder, DPR uses a pretrained BERT encoder to encode documents and never updates them.

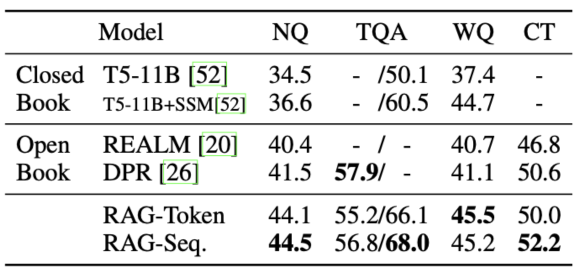
For the decoder side, RAG uses a BART[5] decoder that takes the concatenated query and retrieved document as the input and generates the an answer to the query. Training objective is to maximize the marginal likelihood of the ground-truth targets given input pair (x_j, y_j).

$$\sum_j -\log p(y_j|x_j)$$

The Experiment results show that RAG outperforms DPR and REALM in Open-Domain QA, and generating the entire sequence according to one document(RAG-Seq.) slightly outperforms generating each token individually from different documents(RAG-Token). In Factual Verification(FEVER) datasets, retrieving using BM25 outperforms a neural retriever. The authors suggests that "perhaps since FEVER claims are heavily entity-centric and thus well-suited for word overlap-based retrieval."[6].

3.1 My Thoughts

As I have mentioned before, BERT is not suited to generate spans that answers the query since it is a autoencoding model. It performs the best classifying words and sentences. RAG confirms my intuition that



	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [52]	34.5	- /50.1	37.4	-
	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open Book	REALM [20]	40.4	- / -	40.7	46.8
	DPR [26]	41.5	**57.9**/ -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	**45.5**	50.0
	RAG-Seq.	**44.5**	56.8/**68.0**	45.2	**52.2**

Figure 3: EM scores on OpenQA benchmarks, copied from [6]

using a Encoder-Decoder structure outperforms BERT decoders in QA generations. The results of using another autoregressive model(T5[7], GLM[8]) as the decoder should be promising as well.

References

[1] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`. URL: `https://aclanthology.org/N19-1423`.

[2] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. "Latent Retrieval for Weakly Supervised Open Domain Question Answering". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6086–6096. DOI: `10.18653/v1/P19-1612`. URL: `https://aclanthology.org/P19-1612`.

[3] Kelvin Guu et al. "REALM: Retrieval-augmented language model pre-training". In: *arXiv preprint arXiv:2002.08909* (2020).

[4] Vladimir Karpukhin et al. "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. DOI: `10.18653/v1/2020.emnlp-main.550`. URL: `https://www.aclweb.org/anthology/2020.emnlp-main.550`.

[5] Mike Lewis et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: `10.18653/v1/2020.acl-main.703`. URL: `https://aclanthology.org/2020.acl-main.703`.

[6] Patrick S. H. Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *NeurIPS*. 2020. URL: `https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html`.

[7] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: `http://jmlr.org/papers/v21/20-074.html`.

[8] Zhengxiao Du et al. "GLM: General Language Model Pretraining with Autoregressive Blank Infilling". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 320–335. URL: `https://aclanthology.org/2022.acl-long.26`.