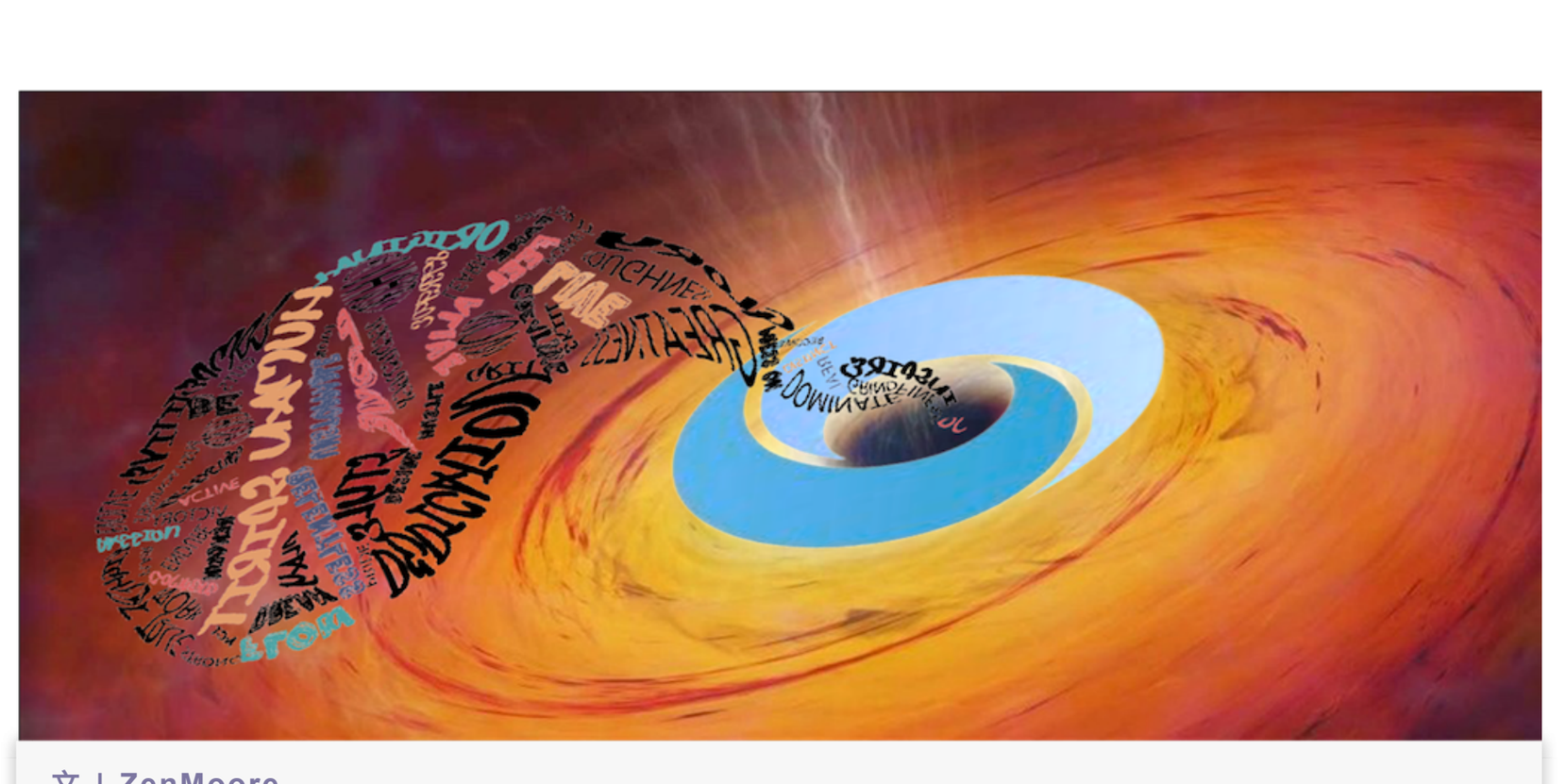
以4%参数量比肩GPT-3！Deepmind 发布检索型 LM，或将成为 LM 发展新趋势！？

原创 ZenMoore 夕小瑶的卖萌屋 2022-01-12 12:05



文 | ZenMoore
编 | 小轶

GPT3[1] 一声枪响，给 NLP 带来了大模型风潮。这么长时间过来，无论是中文还是英文，模型越做越大。当然，这确实是符合逻辑的，因为如果以人脑为向导的话，那么多神经元，不得不需要大参数量来模拟。但是，**过于关注"大"本身，是存在很多问题的**：一方面，模型规模增大带来的性能增益逐渐饱和，让我们不得不思考"精"这个维度；另一方面，大模型实在臃肿，在部署成本、下游任务适配、绿色、边缘化等等方面，有着难以解决的劣势。

因此，大模型发展至今，我们同样需要重点思考的，是**如何把模型做精做强，如何把模型轻量化而效果不减！** 毕竟，模型之大有涯，而知也无涯！以有涯随无涯，殆已！

当然，2021 年出现了很多轻量化模型相关的工作，他们的 Motivation 基本都是采用一系列技巧，把模型的 size 减下来，但是 performance 依旧不输给 GPT3、Megatron[2] 等等超大模型。例如：使用 Prompting 技术的 T0 模型[3]；使用知识增强、训练策略改进、压缩蒸馏、Prompting 等一系列方案的中文孟子模型[4] 等等。

DeepMind 最近也入局了 NLP 模型，上来就是一套组合拳，总计三篇论文：

1. 280B 参数的 Transformer 语言模型：Gopher[5]
2. 大模型的伦理与社会风险研究[6]
3. 检索增强的自回归语言模型：Retro

我们重点聊一聊第三篇：使用**检索增强**的方式，不仅减小了模型的参数量，而且效果也非常能打！因此不失为**模型轻量化**的又一条路：把模型做成 Open System ！

论文标题:
Improving language models by retrieving from trillions of tokens

作者机构:
DeepMind

论文链接:
https://arxiv.org/pdf/2112.04426.pdf

方法

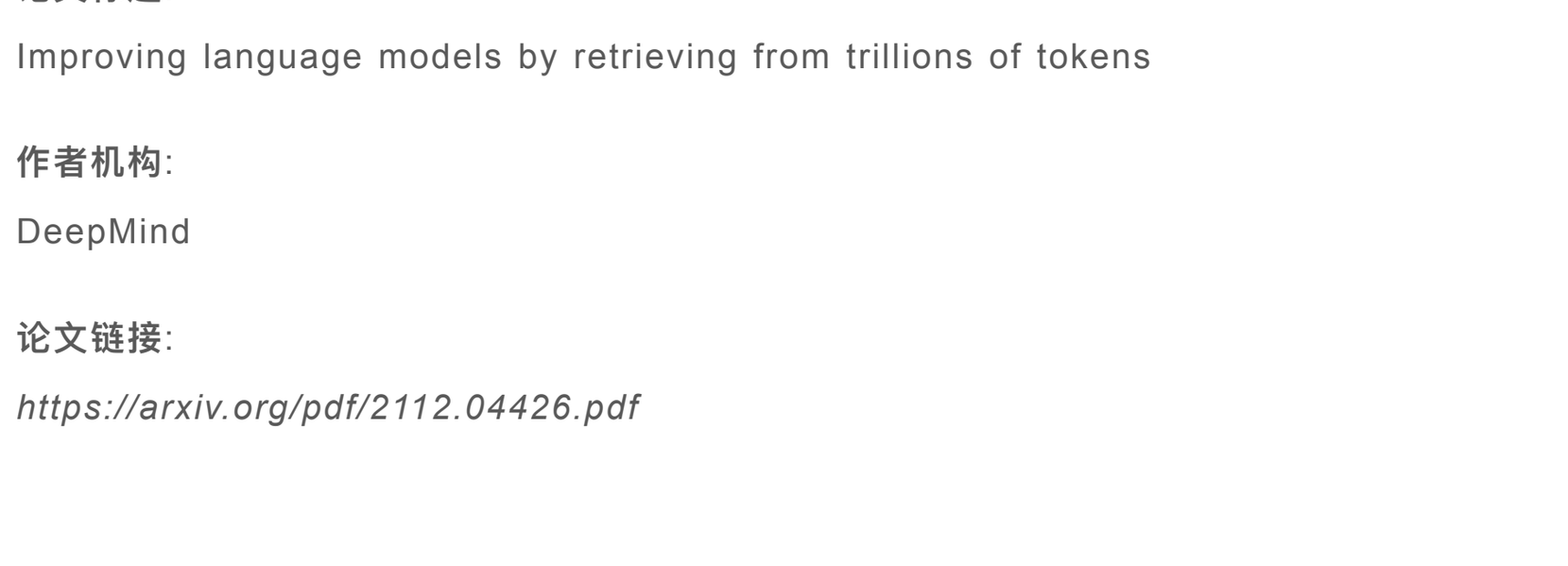


Figure 2 | **Retro architecture.** *Left:* simplified version where a sequence of length $n = 12$ is split into $l = 3$ chunks of size $m = 4$. For each chunk, we retrieve $k = 2$ neighbours of $r = 5$ tokens each. The retrieval pathway is shown on top. *Right:* Details of the interactions in the Cca operator. Causality is maintained as neighbours of the first chunk only affect the last token of the first chunk and tokens from the second chunk.

1. 检索增强的自回归语言模型

从输入开始，首先将输入文本均分为 l 个块(chunk)，每个 chunk 总共 m 个 tokens，即输入文本长度为 $n = ml$. 这些 chunk 分别 记为 $C_1, \dots, C_u, \dots, C_l$. 对每个 C_u，从 Rertieval Dataset 召回 k 个 neighbours，记为 $RET_{\mathcal{D}}(C_u)$ 或者 RET.

因此，目标函数，即序列的对数似然公式如下：

$$L(X \mid \theta, \mathcal{D}) \triangleq \sum_{u=1}^{l} \sum_{i=1}^{m} l_\theta(x_{(u-1)m+i} \mid (x_j)_{j<(u-1)m+i}, (RET_{\mathcal{D}}(C_{u'}))_{u'<u})$$

需要注意的是，$RET(C_1) = \emptyset$.

整个模型的结构是基于 Transformer Encoder-Decoder 的。

2. kNN 检索

然后介绍 Frozen kNN Retriever 部分。

首先有一个事先搜集好的 Retrieval Database（数据来源：MassiveText[5]），总共有 5T tokens 的数据量。将他们以键值对的形式一个 chunk 一个 chunk 地存储起来（不得不说，DeepMind 还是壕啊...）：

- 值：包含两个组分，记为 $[N, F]$, 其中，N 是 text token chunk，F 是这个 token chunk 在数据源文档中的接续文本 (continuation)
- 键：即对应的 BERT embedding，记为 $BERT(N)$ (这里的 BERT 是训练好的，参数不会改变)。

这样，计算输入文本 chunk 的 BERT embedding，将其与 Retrieval Database 中的各个键计算 L_2 距离，作为度量，就可以得到这个输入文本 chunk 的 k 个 nearest neighbours.

3. Retro Model

Algorithm 1: Overview of Retro model architecture.

Hyperparam: P and P_{enc}, indices of layers with cross-attention in the decoder and encoder respectively
Hyperparam: L and L_{enc}, number of decoder layers and number of encoder layers.
Input: $X \in \mathbb{V}^n$: sequence of tokens. $(\text{Ret}(C_u))_{1 \leqslant u \leqslant l}$: the retrieved neighbours
Output: $O \in \mathbb{R}^{n \times |\mathbb{V}|}$: the output logits

```
def Encoder(Ret(C_u)_{1≤u≤l}, H):
    (H_u)_{u∈[1,l]} ← Split(H)
    for j ∈ [1,k], u ∈ [1,l] do // Encoder shared across neighbours and chunks
        E_u^j = Emb_enc(Ret(C_u)^j) // May be shared with the decoder Emb
        for p' ∈ [1, L_enc] do
            E_u^j ← Attn_enc(E_u^j) // Bi-directional attention
            if p' ∈ P_enc then
                E_u^j ← Ca_enc(E_u^j, H_u)
            E_u^j ← Ffw_enc(E_u^j)
    return E

H ← Emb(X)
for p ∈ [1, L] do
    H ← Attn(H) // Causal attention
    if p = min(P) then
        // The neighbour Encoder is conditioned with the decoder activations of
        the last layer before the first cross-attention
        E = Encoder(Ret(C_u)_{1≤u≤l}, H)
    if p ∈ P then
        H ← Cca(H, E)
    H ← Ffw(H)
O ← Read(H)
```

▲Retro 算法流程

接下来介绍 Retro 模型的主体部分，包含 L_{enc} 层 Encoder，以及 L 层 Decoder.

首先记输入文本的第 u 个 chunk 的 activation 为 H_u.

在 Decoder 部分，定义一个整数集合 $P \subseteq [1, L]$，决定哪些层需要使用 Retro-block，然后其余层均使用 Transformer 原版 Decoder 层即可。即：

$$\forall i \in P, \text{layer i is } RETRO(H, E) \triangleq FFW(CCA(ATTN(H), E))$$
$$\forall i \notin P, \text{layer i is } LM(H) \triangleq FFW(ATTN(H))$$

同样地，在 Encoder 部分，也有一个对应的 P_{enc}.

我们这里不聊 Transformer 原版的 Encoder 或者 Decoder 层，只聊新的 Decoder (即 Retro-block) 和新的 Encoder.

新的 Encoder：

对于第 u 个 chunk 的 第 j 近邻的 retrieval neighbour，我们使用这个方式对其进行编码：

$$E_u^j = ENCODER(RET(C_u)^j, H_u)$$

这样就能以一种可微分的方式将输入文本的信息融合到 Retrieval Encoder 里面，从而控制 Retrieval Neighbour 的编码。

新的 Decoder：

这个的核心在于 Chunked Cross-Attention(CCA). 首先以 Figure.2 右侧图的方式，将 H 分割为 $l - 1$ 个 attending chunks，分别记为

$$(H_u^+ \triangleq (h_{um+i-1})_{i \in [1,m]})_{u \in [1, l-1]})$$

然后计算 H_u^+ 和 E_u 的 cross-attention，即

$$CCA(H, E)_{um+i-1} \triangleq CA(h_{um+i-1}, E_u), \forall C_u, \forall i \in [1, m]$$

其中，cross-attention 这样计算：$CA(h, Y) \triangleq softmax(YKQ^T h)YV$.

最后，在 Transformer 的实现上，还有两点小小的细节：

1. LayerNorm 替换为 RMSNorm
2. 使用相对位置编码 (relative position encoding)

4. 数据泄露的量化

除了模型上的改进，作者为了让自己的工作更加严谨有说服力，还针对大规模 Retrieval Database 以及训练集常见的数据泄露问题，提出了更加科学的定量分析方法，可圈可点！

这个问题其实非常自然，训练集、测试集都来自互联网，规模大了，测试数据很容易泄露（即测试集数据出现在训练集中），很多工作对这样的问题睁一只眼闭一只眼，但其实，在卖萌屋往期的推文也谈到过，这非常的不严谨！有没有什么解决办法呢？

首先对于每个测试集（或者验证集）的 chunk $C \in \mathcal{C}$，从训练集中召回 10 个 nearest neighbours，计算 C 与这些 neighbours 的最长公共子序列长度，记作 $s \in [0, m]$. 定义 $r(C) = \frac{s}{m} \in [0, 1]$，用来表征测试集(或验证集)中的 chunk 和训练数据的重合程度。另外，再记 C 编码的字节数为 $N(C)$，C 的对数似然为 $l(C)$，对 $r(C)$ 设置一个阈值 α，就可以定义一个新的评估指标 bits-per-bytes(bpb)：

$$\forall \alpha \in [0, 1], C_\alpha \triangleq \{C \in \mathcal{C}, r(C) \leq \alpha\}, bpb(\alpha) \triangleq \frac{\sum_{C \in \mathcal{C}_\alpha} l(C)}{\sum_{C \in \mathcal{C}_\alpha} N(C)}$$

bpb 值越小，模型的效果越好，同时，可以用阈值 α 来控制对数据泄露问题的容忍度，即 α 越小，bpb 越能代表无数据泄露时的模型效果，另外，bpb 的斜率还能表征模型对泄露数据的依赖度 (how much the model exploits evaluation leakage).

5. 与其他检索方法的对比

Table 3 | **Comparison of Retro with existing retrieval approaches.**

	# Retrieval tokens	Granularity	Retriever training	Retrieval integration
Continuous Cache	$O(10^3)$	Token	Frozen (LSTM)	Add to probs
kNN-LM	$O(10^9)$	Token	Frozen (Transformer)	Add to probs
Spalm	$O(10^9)$	Token	Frozen (Transformer)	Gated logits
Dpr	$O(10^9)$	Prompt	Contrastive proxy	Extractive QA
Realm	$O(10^9)$	Prompt	End-to-End	Prepend to prompt
RAG	$O(10^9)$	Prompt	Fine-tuned Dpr	Cross-attention
FiD	$O(10^9)$	Prompt	Frozen Dpr	Cross-attention
Emdr^2	$O(10^9)$	Prompt	End-to-End (EM)	Cross-attention
Retro (ours)	$\mathbf{O(10^{12})}$	**Chunk**	**Frozen (Bert)**	**Chunked cross-attention**

作者在这个表中总结得很明白，就不赘述了（溜...

实验结果

- Baseline 基本是原版的 Transformer，没有使用 Retrieval 进行增强，改动仅限于 RMSNorm 和相对位置编码
- Retro[OFF] 指的是在 evaluation 阶段，Retro 是不带 Retrieval 的
- Retro[ON] 就是上面介绍的完整的 Retro 模型

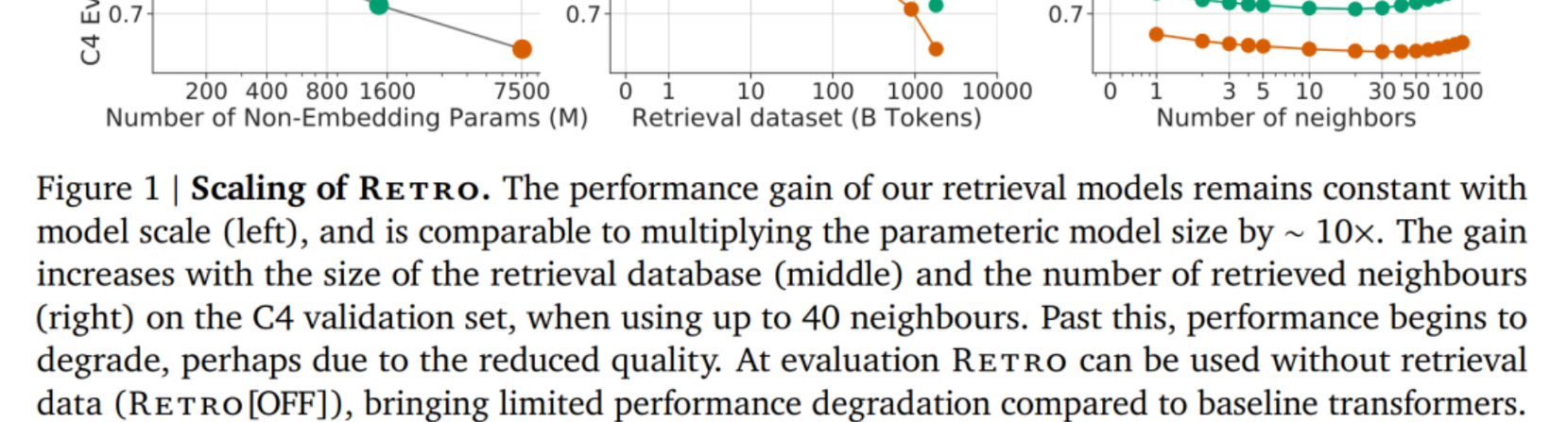


Figure 1 | **Scaling of Retro.** The performance gain of our retrieval models remains constant with model scale (left), and is comparable to multiplying the parameteric model size by ~ 10×. The gain increases with the size of the retrieval database (middle) and the number of retrieved neighbours (right) on the C4 validation set, when using up to 40 neighbours. Past this, performance begins to degrade, perhaps due to the reduced quality. At evaluation Retro can be used without retrieval data (Retro[OFF]), bringing limited performance degradation compared to baseline transformers.

▲注意：纵坐标越小，模型越好

1. 第一张图：蓝色的圆点代表 172M 的 Retro[ON]，红色的三角代表 7.5B 的 Baseline，这两个效果差不多，**证明了 Retrieval 确实可以在保证模型效果的前提下，缩减模型的参数量**
2. 增大 Retrieval Database 的规模，可以有效地提升模型效果
3. Retrieval Neighbours 的数量也会影响模型效果，但是有一个最优点

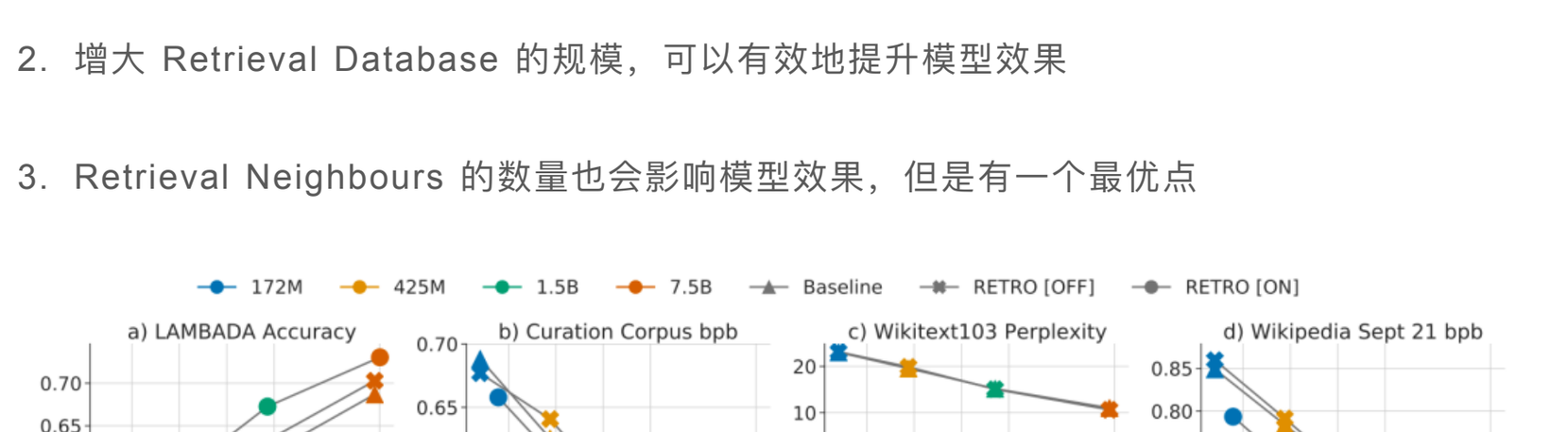


Figure 3 | **Scaling with respect to model size.** (a) LAMBADA top-1 accuracy. (b) Evaluation loss on curation corpus. (c) Perplexity on Wikitext103 valid. (d) Bits-per-byte on selected Wikipedia articles from September 2021.

4. 不同 model size、不同模型在各个数据集上的效果，可以看上面这张图

Table 5 | **Question answering results.** Exact match accuracy on Natural Questions.

Model	Test Accuracy
Realm (Guu et al., 2020)	40.4
Dpr (Karpukhin et al., 2020)	41.5
RAG (Lewis et al., 2020)	44.5
Emdr^2 (Sachan et al., 2021)	52.5
FiD (Izacard and Grave, 2021)	51.4
FiD + Distill. (Izacard et al., 2020)	**54.7**
Baseline 7B (closed book)	30.4
Retro 7.5B (DPR retrieval)	45.5

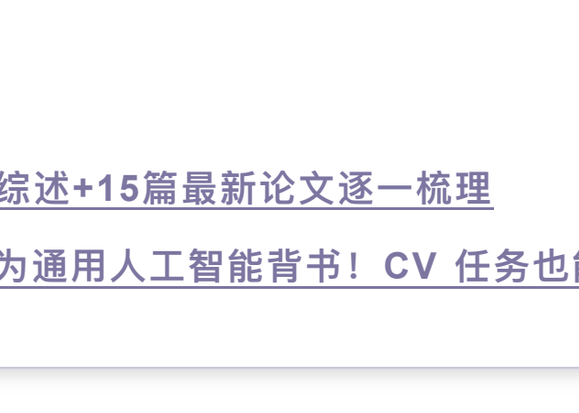
5. 这里挑一个 retrieval intensive 的下游任务——Question Answering. 可见，Retro 仍然战胜了不少模型。（至于为什么比 FID 等[7]略逊一筹，主要是因为 FID 等使用的是 T5-style model[8]，这些模型更加依赖于 encoder output，而这是影响 QA 性能的关键）
6. baseline 模型可以快速灵活地 finetuned 为 Retro 模型，同时效果与 trained from scratch 的模型相比，基本相同。这个结论感觉还挺重要的，**有了这个结论，我们就能够非常灵活地将现有的模型进一步精调成检索型模型，而不需要重新进行漫长的预训练。**
7. 使用作者这种数据泄露问题的定量分析方法，可以证明 Retro 模型的效果增益，基本和数据泄露没有关系。另外，也提倡学界、业界重视起这个问题来，让实验更加严谨！！！

写在最后

DeepMind 很善于从人类身上汲取创造智能的灵感，这一波的灵感是：**人类学习的过程，不仅仅是对当下知识的整合，还包括对记忆的检索，甚至包括对学习资料的检索**。那么，这样一个检索型的语言模型，是非常自然的想法。

另外，虽然作者进行这个工作的初衷并不一定是模型轻量化，但是从它 10x 的参数缩减量来看，确确实实给轻量化提供了一个新的思路。无论是为了 training efficient，还是为了 green，模型的轻量化任重而道远！正如孟子模型的开篇语引用的话一样："以力服人者，非心服也,力不赡也。权,然后知轻重;度,然后知长短。" **把模型做精做强，也是我们应该考虑的核心问题。**

笔者还有一点思考是，当下的语言模型大多是 closed system ：输入数据训练，训练完之后"包裹"起来使用。但是，封闭式系统就意味着，在使用模型进行 inference 的时候，不能"查资料"，是"闭卷考试"，这样真的合理吗？我们都知道，对人类来说，这个世界不会被排除在外，我们一直都在做"开卷考试"：写作时，难免查一查词典，翻一翻名著；编文案时，难免上网找一找灵感，看一看模板。相信对于机器来说，也更需要一个 open system，毕竟 "知也无涯"，而 "模型之大有涯"！以有涯随无涯，殆已！



萌屋作者：ZenMoore
来自北航中法的本科生，数学转码 (AI)，想从 NLP 出发探索人工认知人工情感的奥秘... 个人主页是 zenmoore.github.io，知乎 ID 是 ZenMoore，微信号是 zen1057398161，嘤其鸣矣，求其友声！

作品推荐

1. 一文跟进Prompt进展！综述+15篇最新论文逐一梳理
2. 图灵奖大佬+谷歌团队，为通用人工智能背书！CV 任务也能用 LM 建模！

参考文献

[1]. Language Models are Few-Shot Learners: https://arxiv.org/abs/2005.14165

[2]. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism: https://arxiv.org/pdf/1909.08053.pdf

[3]. Multitask Prompted Training Enables Zero-Shot Task Generalization: https://arxiv.org/abs/2110.08207

[4]. Mengzi: Towards Lightweight yet Ingenious Pre-trained Models for Chinese:https://arxiv.org/pdf/2110.06696.pdf

[5]. Scaling Language Models: Methods, Analysis & Insights from Training Gopher: https://arxiv.org/abs/2112.11446

[6]. Ethical and social risks of harm from Language Models: https://arxiv.org/abs/2112.04359

[7]. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering: https://arxiv.org/pdf/2007.01282.pdf

[8]. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer: https://arxiv.org/abs/1910.10683

[9]. Multitask Prompted Training Enables Zero-Shot Task Generalization: https://arxiv.org/abs/2110.08207