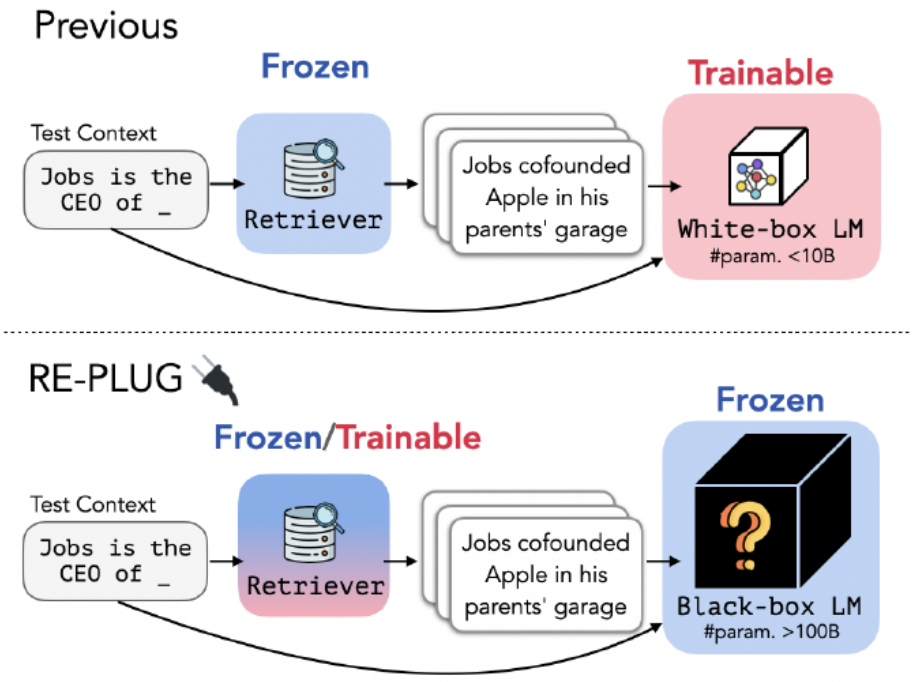
RePLUG

1. 解决什么问题（动机）

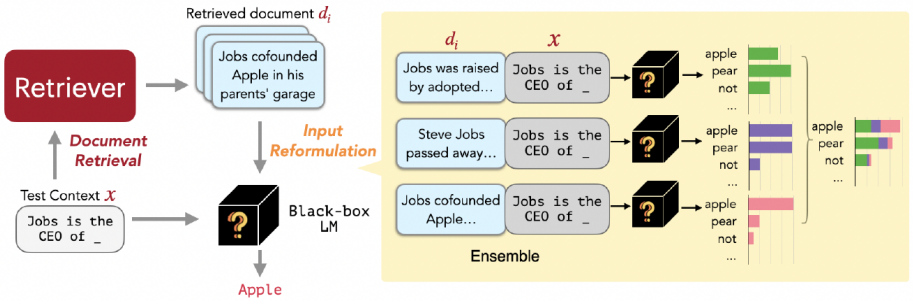
- 先前的工作需要训练或者微调LLM来适配检索的结果
  - 预训练：RETRO，使用检索的知识来微调encoder-decoder模型
  - 微调：Atlas，使用检索的知识库来进行预训练，用于改进encoder-only结构
  - difficult to be applied to very large LMs
  - 问题：针对大于100B的LLM来说
    - 1，大部分都是闭源的，例如GPT3、Codex、Yuan1.0，这些LLM很难结合检索的知识来进行训练优化
    - 2，开源的大模型，例如OPT-175B、BLOOM-176B等训练资源消耗巨大
- 使用KNN-LM将检索文件的token加入到模型推理侧，不用更新LLM，但是需要知道LLM的embedding模块，才可以进行整合
  - many best-in-class LLMs can only be accessed through APIs

2. 如何解决

- 将LLM看作一个黑盒，微调检索器



- 具体训练框架



  - REPLUG训练步骤
    - 第一步：计算检索的文件与query的余弦相似度

$$s(d, x) = \cos(\mathbf{E}(d), \mathbf{E}(x))$$

- 第二步：计算检索文档的归一化得分-作为权重

$$\lambda(d, x) = \frac{e^{s(d,x)}}{\sum_{d \in \mathcal{D}'} e^{s(d,x)}}$$

- 第三步：计算模型的预测损失，乘以对应的权重

$$p(y \mid x, \mathcal{D}') = \sum_{d \in \mathcal{D}'} p(y \mid d \circ x) \cdot \lambda(d, x),$$

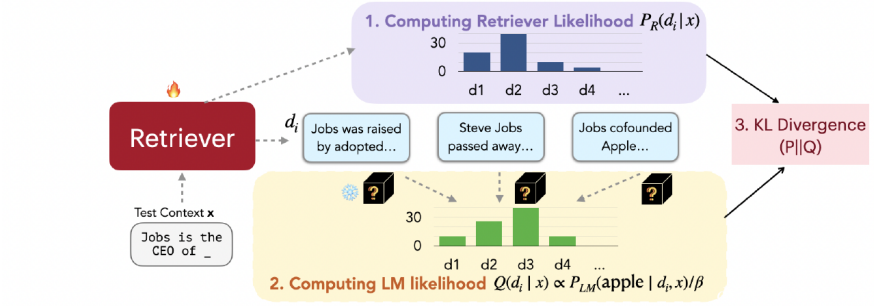


Figure 3. **REPLUG LSR training process (§4).** The retriever is trained using the output of a frozen language model as supervision signals.

- REPLUG-LSR训练的步骤：一共四个
  - 第一步：先使用检索到的文档计算检索的概率分布

$$P_R(d \mid x) = \frac{e^{s(d,x)/\gamma}}{\sum_{d \in \mathcal{D}'} e^{s(d,x)/\gamma}}$$

  - 第二步：使用LLM模型计算检索的文件得分

$$Q(d \mid x, y) = \frac{e^{P_{LM}(y|d,x)/\beta}}{\sum_{d \in \mathcal{D}'} e^{P_{LM}(y|d,x)/\beta}}$$

    - D'表示k个检索的文档，每个query需要计算k次，k表示检索的文档数目
  - 第三步：通过最小化两个分布的差异来更新检索器的参数

$$\mathcal{L} = \frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} KL\Big(P_R(d \mid x) \parallel Q_{\text{LM}}(d \mid x, y)\Big),$$

  - 第四步：每T steps重新计算文件的embeddings并且使用新的embeddings重新保存数据库（更新索引）
- 训练的tricks
  - 为了避免LLM的限制长度对检索结果的影响，提出一种简单的策略，就是为每一个检索的知识+query生成一个回答
  - 预先计算文件的embedding，并且创建Faiss索引

3. 实验结果

- 训练数据
  - 训练数据集，使用Pile training data，采样了800k的序列，每个序列长度为256，使用前128预测后128，从该数据集中抽取36M的文件长度为128token作为知识库-注意需要与训练集进行去重
- 参数配置
  - 给定一个query x，选取top 20的文档，然后计算KL损失，温度系数为0.1，使用Adam优化器，batch为64，热启动设置为0.1，每3k steps重新计算文件的embedding，微调检索器一共25k steps
- 整体效果

REPLUG: Retrieval-Augmented Black-Box Language Models

Model		# Parameters	Original	+ REPLUG	Gain %	+ REPLUG LSR	Gain %
GPT-2	Small	117M	1.33	1.26	5.3	1.21	9.0
	Medium	345M	1.20	1.14	5.0	1.11	7.5
	Large	774M	1.19	1.15	3.4	1.09	8.4
	XL	1.5B	1.16	1.09	6.0	1.07	7.8
GPT-3 (black-box)	Ada	350M	1.05	0.98	6.7	0.96	8.6
	Babbage	1.3B	0.95	0.90	5.3	0.88	7.4
	Curie	6.7B	0.88	0.85	3.4	0.82	6.8
	Davinci	175B	0.80	0.77	3.8	0.75	6.3

Table 1. **Both REPLUG and REPLUG LSR consistently enhanced the performance of different language models.** Bits per byte (BPB) of the Pile using GPT-3 and GPT-2 family models (Original) and their retrieval-augmented versions (+REPLUG and +REPLUG LSR. The gain % shows the relative improvement of our models compared to the original language model.

- MMLU实验结果

Model	# Parameters	Humanities	Social.	STEM	Other	All
Codex	175B	74.2	76.9	57.8	70.1	68.3
PaLM	540B	77.0	81.0	55.6	69.6	69.3
Flan-PaLM	540B	-	-	-	-	72.2
Atlas	11B	46.1	54.6	38.8	52.8	47.9
Codex + REPLUG	175B	76.0	79.7	58.8	72.1	71.4
Codex + REPLUG LSR	175B	76.5	79.9	58.9	73.2	71.8

Table 2. **REPLUG and REPLUG LSR improves Codex by 4.5% and 5.1% respectively.** Performance on MMLU broken down into 4 categories. The last column averages the performance over these categories. All models are evaluated based on 5-shot in-context learning with direct prompting.

4. 启发（可以借鉴的东西）

- 借鉴其思想来优化检索器
- 在限制token的长度的情况下，可以采用类似的方法，针对每一个检索的文档+一个query构成一条数据进行生成（需要生成k次）

5. 参考资料

- 论文：https://arxiv.org/pdf/2301.12652
- RAG汇报：https://yach-doc-shimo.zhiyinlou.com/docs/1lq7MZORpxFmjxAe/ <RAG技术汇报>