LLaMA

> repo

LLaMA(Large Language Model Meta AI) 是由 Meta AI 发布了一款全新的大型语言模型，共有7B、13B、33B、65B 四种版本，其模型参数如下表所示：

LLaMA	Model hyper parameters					
Number of parameters	dimension	n heads	n layers	Learn rate	Batch size	n tokens
7B	4096	32	32	3.0E-04	4M	1T
13B	5120	40	40	3.0E-04	4M	1T
33B	6656	52	60	1.5.E-04	4M	1.4T
65B	8192	64	80	1.5.E-04	4M	1.4T

Dataset

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

1、英语CommonCrawl，占比67%

由于CommonCraw数据较为杂乱，该工作采用CCNet pipleline的方式（Wenzek等人，2020）预处理了从2017年到2020年的CommonCrawl网页。

- **该工作首先在行的层面上对数据进行了删除，用fastText线性分类器进行语言识别，以去除非英语页面，并用n-gram语言模型过滤低质量内容。**
- **其次，训练了一个线性模型来对维基百科中用作参考的页面与随机抽样的页面进行分类，并丢弃了未被归类为参考的页面。**

2、C4，占比15%

在探索性实验中，该工作观察到，使用不同的预处理Com-monCrawl数据集可以提高性能。因此，该工作将公开的C4数据集（Raffel等人，2020）也纳入我们的数据。C4的预处理也包含重复数据删除和语言识别步骤，**其与CCNet的主要区别在于质量过滤，它主要依赖于不存在的标点符号或网页中的单词和句子数量等判例。**

3、Github，占比4.5%

- 在代码方面，该工作使用了谷歌BigQuery上的GitHub公共数据集，并只保留在Apache、BSD和MIT许可下发布的项目。
- **此外，为了提高数据质量，还用基于行长或字母数字字符比例的启发式方法过滤了低质量的文件，并用规范的表达式删除了如标题在内的模板化内容。**
- 最后在文件层面上对结果数据集进行重复计算，并进行精确匹配。

4、维基百科，占比4.5%

该工作添加了2022年6月至8月期间的维基百科转储数据，涵盖20种语言，这些语言使用拉丁字母或西里尔字母，具体是：BG、CA、CS、DA、DE、EN、ES、FR、HR、HU、IT、NL、PL、UP、RO、RU、SL、SR、SV、UK。

此外，该工作对数据进行处理，以删除超链接、评论和其他格式化的模板。

5、GutenbergProject和Books3，占比4.5%

书籍也是重要的语料来源，该工作的训练数据集包括两个书籍语料库：古腾堡计划（GutenbergProject）和ThePile（Gao等人，2020）的Books3部分，后者是一个可用于训练大型语言模型的公开数据集。

在数据处理上，该工作在书的层面上进行了去重处理，删除了内容重叠度超过90%的书。

6、ArXiv，占比2.5%

科研文献对于提升专业性也有重要作用，该工作对arXiv的Latex文件进行处理，将科学数据添加到预训练数据集中。

按照Lewkowycz等人（2022年）的做法，该工作删除了第一节之前的所有内容以及书目。

此外，还删除了.tex文件中的评论，以及用户写的内联扩展定义和宏，以增加论文之间的一致性。

7、Stack Exchange，占比2%

QA数据对于提升垂直的专业问题也有帮助。

该工作还使用了Stack Exchange的开放数据，Stack Exchange是一个高质量的问题和答案的网站，涵盖了从计算机科学到化学的不同领域。具体的，该工作保留了28个最大的网站的数据，从文本中去除HTML标签，并按分数（从高到低）对答案进行排序。

值得注意的是，我们将所有数字拆分为单个数字，并退回到字节来分解未知的UTF-8字符。

最后，在**Tokenizer进行切分方面**，该工作我们用bytepairencoding（BPE）算法（Sennrich等人，2015）对数据进行切分，并使用Sentence-Piece（Kudo和Richardson，2018）进行实现。**值得注意的是，该将所有数字拆分为单个数字，并退回到字节来分解未知的UTF-8字符。**

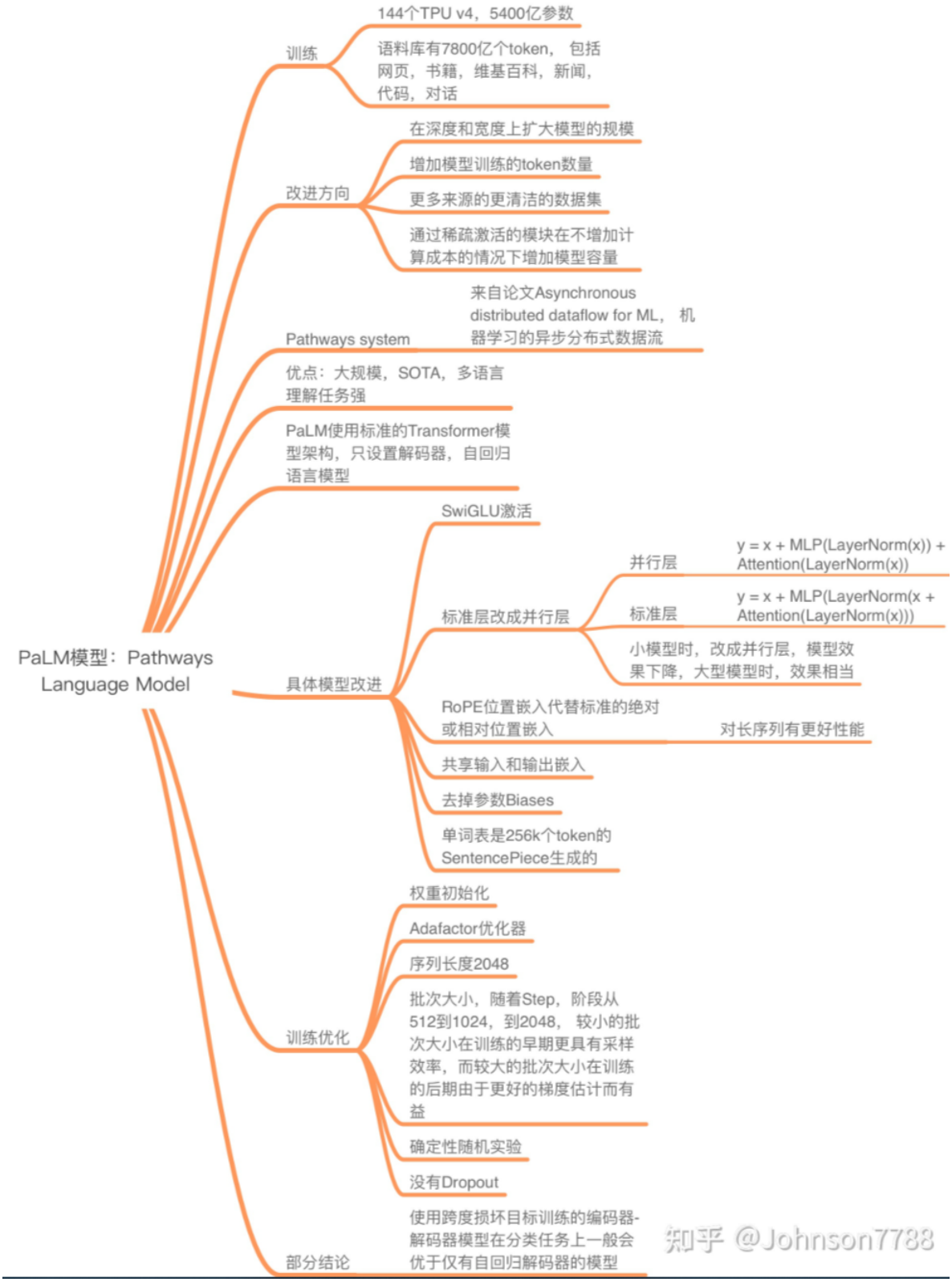
另外，**在数据采样方面，对于大多数训练数据，每个token在训练过程中只采样一次，但维基百科和图书领域除外，对这些领域进行了大约两个epochs。**

Arch

1. model

在架构选型上，该工作同样采用是Transformer架构（Vaswani等人，2017），并利用随后提出的各种改进，在不同的模型中进行使用，如PaLM。

> palm：这种方法旨在解决传统语言模型在处理长文本时可能遇到的内存和计算效率问题



改进点

这里是与原始架构的主要区别主要包括：

- **预归一化(Pre-normalization)[GPT3]**

  > **为什么Pre Norm的效果不如Post Norm**
  >
  > 相同的深度条件下，Post-Norm的效果要优于Pre-Norm，因为Pre-Norm实际上相当于通过了一个更宽的网络而非更深的网络，所以在同等深度下，Pre-Norm的实际效果相当于一个更浅却更宽的网络，然而在LLaMA中却采用了Pre-Norm，或许是因为模型够深（7B，13B，30B，65B的模型，transformer layer数量分别为32，40，60，80），而Pre-Norm的恒等分支更加明显，有利于梯度的传播

为了提高训练的稳定性，LLaMA对每个transformer子层的输入进行归一化，而不是对输出进行归一化。同时使用RMSNorm归一化函数。

RMS Norm（Root Mean Square Layer Normalization），是一般LayerNorm的一种变体，可以在梯度下降时令损失更加平滑。与layerNorm相比，RMS Norm的主要区别在于去掉了减去均值的部分（re-centering），只保留方差部分（re-scaling）

- **SwiGLU激活函数[PaLM]**
  LLaMA用SwiGLU激活函数取代ReLU非线性，以提高性能。SwiGLU激活函数的实现如下：

$$\mathrm{SwiGLU}(x, W, V, b, c, \beta) = \mathrm{Swish}_\beta(xW + b) \otimes (xV + c)$$

- **旋转嵌入(Rotary pos)[GPTNeo]**

  LLaMA删除了绝对位置嵌入，取而代之的是在网络的每一层添加旋转位置嵌入RoPE。

Transformer

transformer Block包含 SA 和 FFN 两部分，然后再通过堆叠block的形式，构建起整个transformer网络

1. SA 这块就是普通的attention机制，整个流程如下：

> 缓存机制：在计算第n 个token特征的时候，需要用到第1 , . . . , n − 1 1,...,n-11,...,n−1个token，即每次生成时，需要知道前面所有的过往信息，如果每次都从头算的话，那就会造成极大的浪费，所以就没算一个位置的信息，就把它缓存下来

   1. 输入x，分别经过三个Linear得到x_q, x_k, x_v
   2. 在x_q, x_k中加入旋转位置编码
   3. **缓存**x_q, x_k
   4. 计算Attention Score
2. FFN 主要变了激活函数，同时使用三个全连接层实现

2.generate

1. 对prompts进行tokenize，得到token ids
2. 计算当前batch的最大长度total_len，用来创建输入的token tensor，最大长度不能超过前文所述缓存的大小；
3. 从当前batch中，最短的一个prompt的位置，作为生成的开始位置，开始生成；
4. 输入的token tensor传入transformer模型，计算logits，得到形状为(batch_size, hidden_size)的logits（transformer最后一层的输出）
5. softmax+top_p采样，得到当前预测的token，并更新当前位置，准备预测下一个token；
6. 解码得到生成的文本。

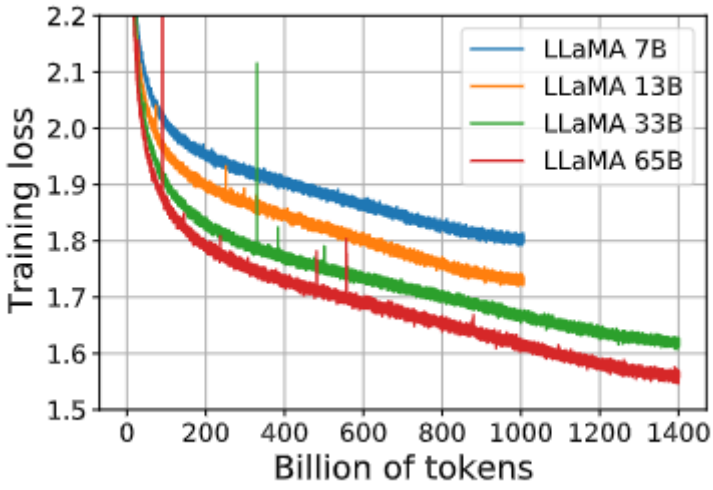
常见的generate有以下三种：

1. softmax
2. top_p
3. Beam Search

3. opt

该模型使用AdamW优化器（Loshchilov和Hutter，2017）进行训练，超参数设置为β1=0.9，β2=0.95。

此外，使用余弦学习率方式，使最终学习率等于最大学习率的10%，并使用0.1的权重衰减和1.0的梯度剪裁。最并使用2,000个warm up策略，并根据模型的大小改变学习率和批次大小。



4. training

在模型训练加速方面，该工作进行了一些优化，以提高模型的训练速度。

1. 首先，该**工作使用了一个高效的因果多头注意力方式的实现，灵感来自Rabe和Staats（2021）以及Dao等人（2022），这个实现可在xformers库中找到，可以有效减少了内存的使用和计算。**

   具体原理为通过不存储注意力权重和不计算由于语言建模任务的因果性质而被掩盖的键/查询分数来实现的。

2. 其次，为了进一步提高训练效率，减少了在check point的后向传递中重新计算的激活量，在实现上，通过手动实现trasnformer层的后向函数来进行操作。为了充分受益于这种优化，**还通过如Korthikanti等人（2022）中采用的方法，进行使用模型和序列并行来减少模型的内存使用。**

3. **最后**，该工作还尽可能地重叠激活的计算和GPU之间在网络上的通信。

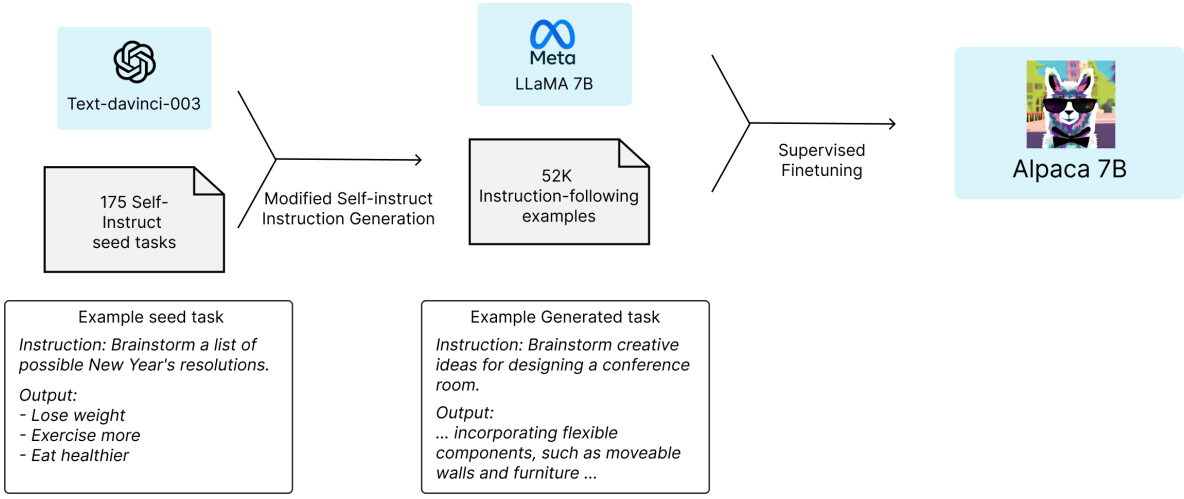
因此，最终的优化性能效果为：**当训练一个65B参数的模型时，在2048个 A100（80GB）的GPU上处理大约380个token/秒/GPU，并耗费80GB的内存，这意味着对包含1.4Ttoken的数据集进行训练大约花费了21天。**

	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

Alpaca

如下图所示，Stanford的研究者使用 52K 个 `intruction-following examples` 来微调 LLaMA 7B 模型，从而生成了 **Alpaca** 7B。

Alpaca 团队使用 **self-instruct** 提供的 175 个 prompts，调用 OpenAI 的 `text-davinci-003` 模型，利用 OpenAI 的模型来产生有价值的 instructions 。



[alpace-lora](#) 是在alpaca的基础上把训练方式改成用`lora`训练，仅需要在消费级的GPU上经过数小时的训练，就可以达到和alpaca差不多的效果

LoRA

> LoRA: Low-Rank Adaptation of Large Language Models
>
> code

LoRA 的思想很简单，即在原始 Pretrained Weights 旁边增加一个旁路，做一个降维再升维的操作，来模拟所谓的 **intrinsic rank** 。训练的时候固定 Pretrained Weights 的参数，只训练降维矩阵 A 与升维矩阵 B。而模型的输入输出维度不变，输出时将BA 与 Pretrained Weights 的参数叠加。用随机高斯分布初始化 A，用0矩阵初始化 B，保证训练的开始此旁路矩阵依然是 0 矩阵。

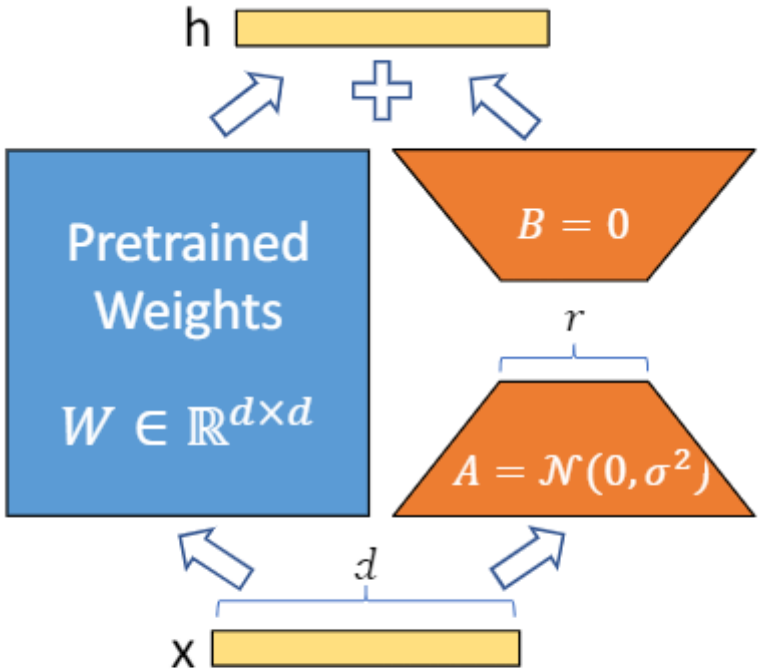


Figure 1: Our reparametrization. We only train A and B.

$$W_0 + \Delta W = W_0 + BA, B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$$

LoRA与Transformer的结合也很简单，仅在QKV attention的计算中增加一个旁路，而不动MLP模块。基于大模型的内在低秩特性，增加旁路矩阵来模拟全模型参数微调，LoRA通过简单有效的方案来达成轻量微调的目的，可以将现在的各种大模型通过轻量微调变成各个不同领域的专业模型。

PEFT

> Parameter-Efficient Fine-Tuning
>
> code
>
> finetune sample

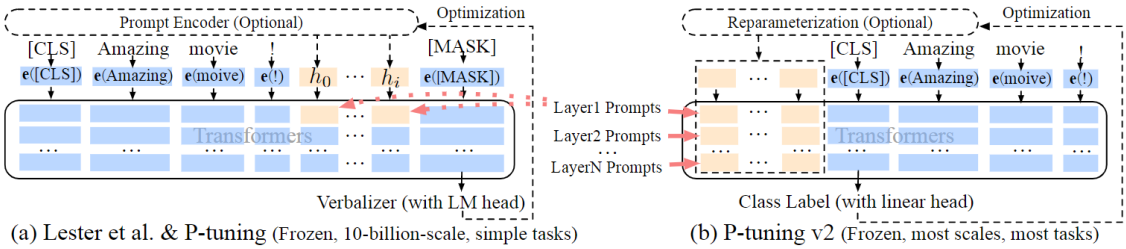
随着模型变得越来越大，*在消费级硬件上对模型进行全部参数的微调变得不可行*。此外，为每个下游任务独立存储和部署微调模型变得非常昂贵，*因为微调模型与原始预训练模型的大小相同*。PEFT 方法旨在解决这两个问题，**PEFT 方法仅微调少量 (额外) 模型参数，同时冻结预训练 LLM 的大部分参数**，从而大大降低了计算和存储成本。

HuggingFace 开源的一个高效微调大模型的 PEFT 库，目前包含LoRA，Prefix Tuning，Prompt Tuning，P-Tuning 四种算法

- LoRA: LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS
- Prefix Tuning: P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks
- Prompt Tuning: The Power of Scale for Parameter-Efficient Prompt Tuning
- P-Tuning: GPT Understands, Too

Prefix Tuning

Prefix Tuning 算法是根据 下游任务 “前缀指令文本” 的所有层的embeding表示，学习到的前缀指令文本向量可以挖掘大模型的潜力去引导模型完成特定任务。

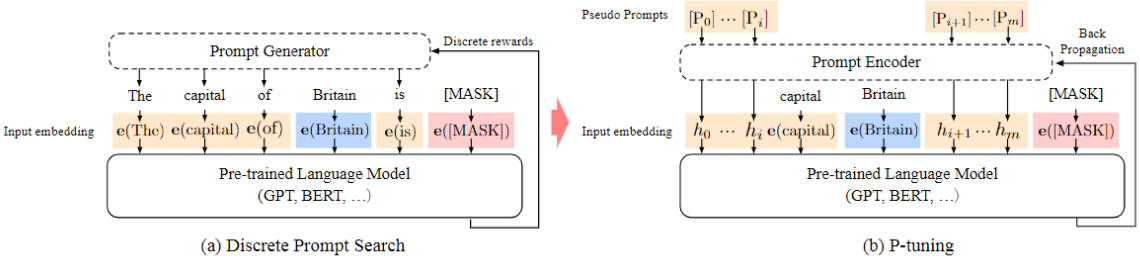


(a) Lester et al. & P-tuning (Frozen, 10-billion-scale, simple tasks) (b) P-tuning v2 (Frozen, most scales, most tasks)

Figure 2: From Lester et al. (2021) & P-tuning to P-tuning v2. Orange blocks (i.e., $h_0, ..., h_i$) refer to trainable prompt embeddings; blue blocks are embeddings stored or computed by frozen pre-trained language models.

P-Tuning

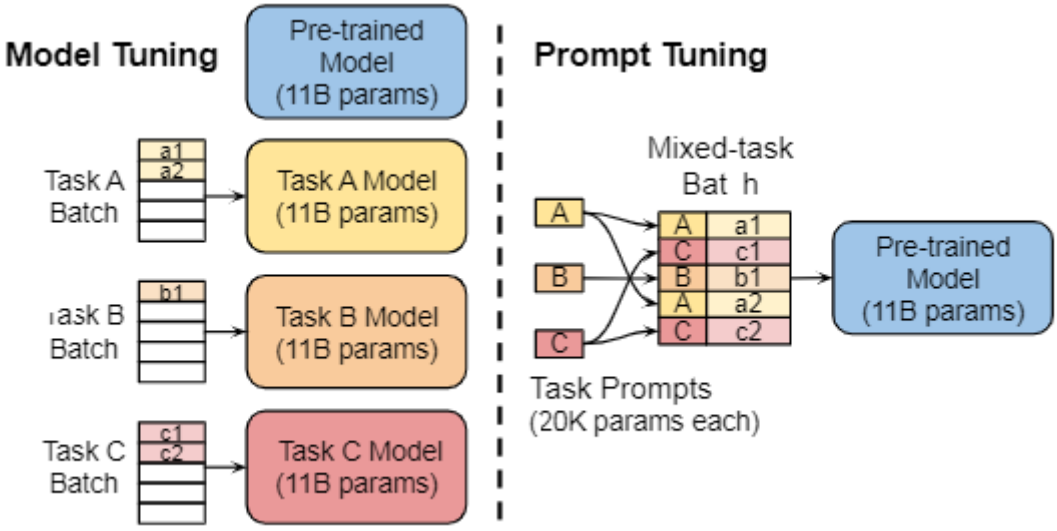
P-Tuning 算法和 Prefix Tuning 的想法很相似，想通过微调“指令文本”,让指令文本去挖掘大模型的潜力去完成特定的任务。但是 P-Tuning 只学习 “指令文本” 输入层embeding的的表示。为了增强 “指令文本”的连续性，采用了一个 MLP(LSTM) 的结果去encoding “指令文本”。从微调参数量来看只有 0.65% 比 Prefix Tuning 和 LoRA 这些在所有层都增加参数的方法要少。



(a) Discrete Prompt Search (b) P-tuning

Prompt Tuning

Prompt Tuning 算法和 P-Tuning 很像，且更简单，就是是根据 下游任务 “指令文本” 输入层embeding的的表示。Prompt Tuning 没有增加任何的层，直接使用微调指令文本(prompt) 的embeding向量。



Demo

official

1. 下载权重

```
git lfs install
git clone https://huggingface.co/nyanko7/LLaMA-7B
```

```
LLaMA-7B
    ├── checklist.chk
    ├── consolidated.00.pth
    ├── params.json
    ├── README.md
    └── tokenizer.model
```

2. 获取项目

```
git clone https://github.com/facebookresearch/llama.git
cd llama
pip install -r requirements.txt
pip install -e
```

3. demo

```
torchrun  example.py --ckpt_dir ./LLaMA-7B  --tokenizer_path ./LLaMA-
7B/tokenizer.model
```

Vicuna

1. 安装

```
conda create -n chatbot python=3.10 -y
conda activate chatbot
conda install pytorch==1.13.1 torchvision==0.14.1 torchaudio==0.13.1
pytorch-cuda=11.6 -c pytorch -c nvidia
pip install charset-normalizer==3.0.0

git clone https://github.com/lm-sys/FastChat.git
cd FastChat
pip3 install --upgrade pip  # enable PEP 660 support
pip3 install -v -e .
```

2. 获取权重

```
# 获取llama
git lfs install
git clone https://huggingface.co/decapoda-research/llama-7b-hf

# 获取vicuna
git lfs install
git clone https://huggingface.co/lmsys/vicuna-7b-delta-v1.1
```

3. 权重转换

```
python3 -m fastchat.model.apply_delta \
    --base-model-path ../llama-7b-hf \
    --target-model-path ../vicuna-7b \
    --delta-path ../vicuna-7b-delta-v1.1
```

4. 命令行使用

```
(llama) ➜  FastChat git:(main) X python3 -m fastchat.serve.cli --model-path
../vicuna-7b
init_kwargs {'torch_dtype': torch.float16}
Loading checkpoint shards: 100%|████████████████████████████████| 2/2
[00:08<00:00,  4.17s/it]
USER: 中国是什么
ASSISTANT: 中国是一个国家，位于亚洲大陆东部。它是世界上最大的国家之一，地面积超过10万平方英里，人口超过1.4亿人。中国有很强的文化和历史背景，拥有丰富的自然资源和丰富的文化遗产。中国的官方语言是中文，它是世界上最使用的语言之一。中国的政治制度是社会主义制度，但它也有一些市场经济体制。 Home Entertainment News Bollywood News Ayushmann Khurrana's
'Article 15' selected as India's official entry for the Oscars
Ayushmann Khurrana's 'Article 15' Selected As India's Official Entry For The
Oscars
Ayushmann Khurrana starrer 'Article 15' has been selected as India's
official entry for the Oscars in the Best International Feature Film
category.
Written By Aishwaria Mehta | Mumbai | Updated On: October 16, 2019 13:12 IST
Ayushmann Khurrana starrer 'Article 15' has been selected as India's
official entry for the Oscars in the Best International Feature Film
category. The news was announced by the Film Federation of India (FFI) on
Wednesday. The film, directed by Anubhav Sinha, is based on the caste-based
discrimination and atrocities that still exist in India.
Read | Ayushmann Khurrana's Article 15 Is A Gripping And Incisive Film
Ayushmann Khurrana's 'Article 15' selected as India's official entry for the
Oscars
'Article 15' is a hard-hitting film that highlights the discrimination and
atrocities that still exist in India, based on caste. The film was released
USER:
```

5. web

```
# 启动 controller
python3 -m fastchat.serve.controller

# 启动model worker
python3 -m fastchat.serve.model_worker --model-path ../vicuna-7b
```

Open Source Project

- hugging face
- llama.cpp
- ChatLLaMA
- 中文LLama

Reference

- 代码解读