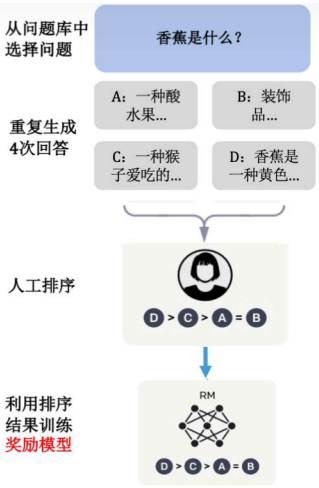


奖励模型训练细节

- GPT-3 architecture
- 6B。175B会不稳定。
- 模型最后替换为投影层，输出标量值。

Loss Function：同一prompt x的会生成多个输出结果，根据人工rank的结果计算reward之间的pair-wise loss。

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

【RLHF】想训练ChatGPT？得先弄明白Reward Model怎么训（附源码）

何枝
如果人人都做有意义的事，那有趣的事该由谁去做呢。
关注他

收起

:排序序列」替代「直...
Loss —— 通过排序序...
:果
:台

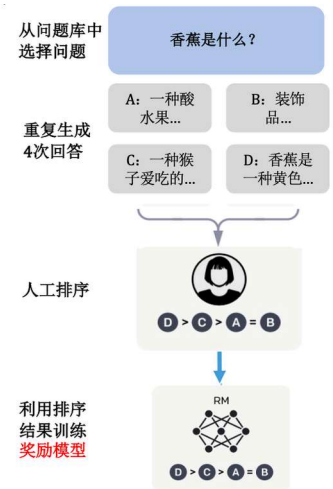
> 完整源码在文末。

在上一篇文章中，我们已经讲解了如何将强化学习（Reinforcement Learning）和语言模型（Language Model）做结合：

https://zhuanlan.zhihu.com/p/595116794
zhuanlan.zhihu.com/p/595116794

但是，示例中我们是使用一个现成的「情绪识别模型」来作为奖励模型（Reward Model）。

在 ChatGPT 中，奖励模型是通过人工标注的「排序序列」来进行训练的，如下图所示：



奖励模型训练细节

- GPT-3 architecture
- 6B。175B会不稳定。
- 模型最后替换为投影层，输出标量值。

Loss Function：同一prompt x的会生成多个输出结果，根据人工rank的结果计算reward之间的pair-wise loss。

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

InstructGPT Reward Model 训练流程

这是什么意思呢？

如上图所示，ChatGPT 并不是直接让人工去标注每一句话的真实得分是多少（尽管模型最终要预测的就是每句话的得分），而是让人去对 4 句话按照好坏程度进行「排序」。

通过这个「排序序列」，模型将会学习如何为每一个句子进行打分。

听起来很绕对吧？

既然最终目的是训练一个句子打分模型，为什么不让人直接打分，而是去标排序序列呢？

今天我们就来好好聊一聊这个非常巧妙的思想。

视频讲解在这里：

ChatGPT训练思路复现实验
10.1 万播放 · 149 赞同 视频

1. 「标注排序序列」替代「直接打分」

大家在曾经考语文的时候，都写过作文吧？

而作文的分数也成为了整个语文考试中不确定性最大的环节。

因为「打分」这个行为的主观性太强，同一篇作文不同的老师可能会打出不同的分数。

为了统一打分标准，通常在阅卷的时候都会制定一系列的规则，例如：主题明确，语句通顺，句子优美等。

但，即便如此，不同老师对「主题明确」和「句子优美」也有着不同的看法。

这就导致我们很难统一所有老师的看法，使得不同人在看到同一篇作文时打出相同的分数。

而标注员在给 ChatGPT 进行标注的时候，就可以看做有很多个「老师」在给模型写的作文「打分」。

因此我们可以看出，直接给生成文本进行打分是一件非常难统一的事情。

如果对于同样的生成答案，有的标注员打 5 分，但有的标注员打 3 分，模型在学习的时候就很难明确这句话究竟是好还是不好。

既然打「绝对分数」很难统一，那我们转换成一个「相对排序」的任务是不是就容易许多呢？

举例来讲，假设今天模型生成了 2 句话：

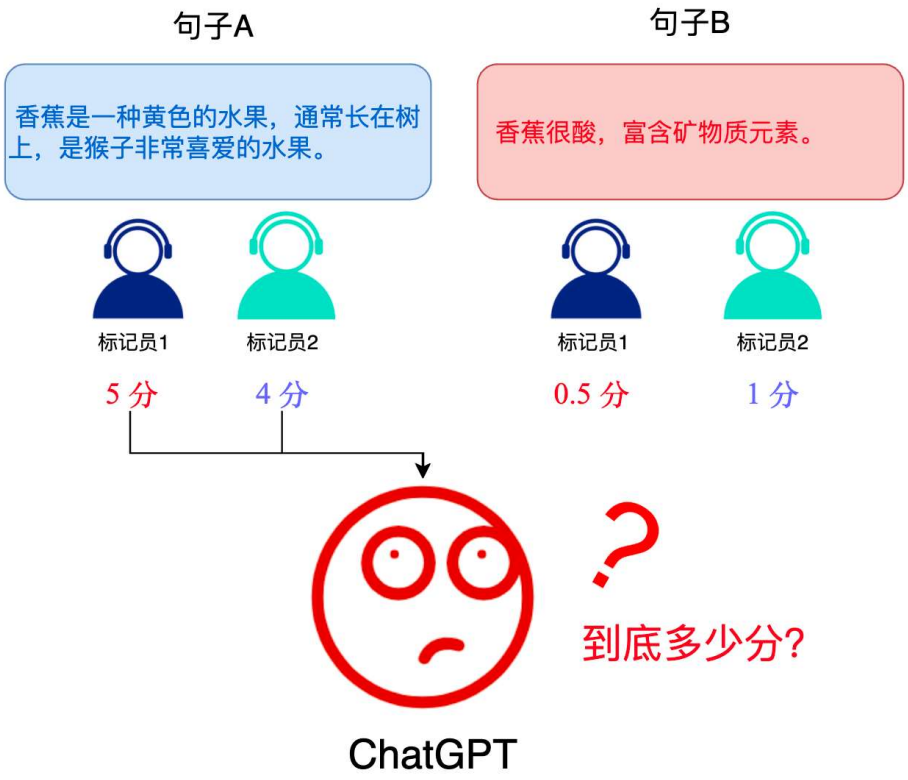
1. 香蕉是一种黄色的水果，通常长在树上，是猴子非常喜爱的水果。
2. 香蕉很酸，富含矿物质元素。

如果让作业员去打分，可能不同人打出来不同的分：

生成句子	得分（标注员1）	得分（标注员2）
香蕉是一种黄色的水果，通常长在树上，是猴子非常喜爱的水果。	4.5	5.0
香蕉很酸，富含矿物质元素。	1.0	0.5

但如果我们只让标注员对这两个答案进行好坏排序，就能得到统一的结果：

生成句子对	排序（标注员1）	排序（标注员2）
A：香蕉是一种黄色的水果，通常长在树上，是猴子非常喜爱的水果。 B：香蕉很酸，富含矿物质元素。	A > B	A > B



「绝对分数」难以统一

「相对排序」容易统一

不难看出，用「相对任务」替代「绝对任务」能够更方便标注员打出统一的标注结果。

那么，「统一」的问题解决了，我们怎么通过「排序序列」来教会模型「打分」呢?

2. Rank Loss —— 通过排序序列学会打分

假定现在有一个排好的序列：A > B > C >D。

我们需要训练一个打分模型，模型给四句话打出来的分要满足 r(A) > r(B) > r(C) > r(D)。

那么，我们可以使用下面这个损失函数：

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

其中，yw 代表排序排在 yl 的所有句子。

用上述例子（A > B > C > D）来讲，loss 应该等于：

```
loss = r(A) - r(B) + r(A) - r(C) + r(A) - r(D) + r(B) - r(C) + ... + r(C) - r(D)
loss = -loss
```

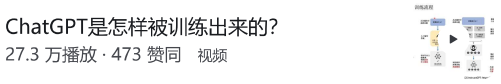
为了更好的归一化差值，我们对每两项差值都过一个 sigmoid 函数将值拉到 0 ~ 1 之间。

可以看到，loss 的值等于排序列表中所有「排在前面项的reward」减去「排在后面项的reward」的和。

而我们希望模型能够「最大化」这个「好句子得分」和「坏句子得分」差值，而梯度下降是做的「最小化」操作。

因此，我们需要对 loss 取负数，就能实现「最大化差值」的效果了。

更详细的解释可以参考下面这个视频中（14:55秒）的例子：



3. 实验结果

这一小节中，我们将尝试通过「排序序列」来学习一个「打分模型」。

首先我们会先准备一份数据集，每一行是一个排序序列（用\t符号隔开）。

排在越前面的越偏「正向情绪」，排在越后面越「负向情绪」。

```
1.买过很多箱这个苹果了，一如既往的好，汁多味甜～	2.名不副实。	3.拿过来居然屏幕有划痕，
1.一直用沙宣的洗发露！是正品！去屑止痒润发护发面面俱到！	2.觉得比外买的稀，好似加了水的
...
```

我们期望通过这个序列训练一个 Reward 模型，当句子越偏「正向情绪」时，模型给出的 Reward 越高。

在 backbone 上，我们选用 ERNIE 作为基准模型，将模型的 pooler_output 接一层 linear layer 以得到一维的 reward：

```
class RewardModel(nn.Module):

    def __init__(self, encoder):
        """
        init func.

        Args:
            encoder (transformers.AutoModel): backbone, 默认使用 ernie 3.0
        """
        super().__init__()
        self.encoder = encoder
        self.reward_layer = nn.Linear(768, 1)            # reward layer 用于映射到 1 维

    def forward(
        self,
        input_ids: torch.tensor,
```

```
        token_type_ids: torch.tensor,
        attention_mask=None,
        pos_ids=None,
    ) -> torch.tensor:
        """
        forward 函数，返回每句话的得分值。

        Args:
            input_ids (torch.tensor✦): (batch, seq_len)
            token_type_ids (torch.tensor): (batch, seq_len)
            attention_mask (torch.tensor): (batch, seq_len)
            pos_ids (torch.tensor): (batch, seq_len)

        Returns:
            reward: (batch, 1)
        """
        pooler_output = self.encoder(
            input_ids=input_ids,
            token_type_ids=token_type_ids,
            position_ids=pos_ids,
            attention_mask=attention_mask,
        )["pooler_output"]                                  # (batch, hidden_size)
        reward = self.reward_layer(pooler_output)           # (batch, 1)
        return reward
```

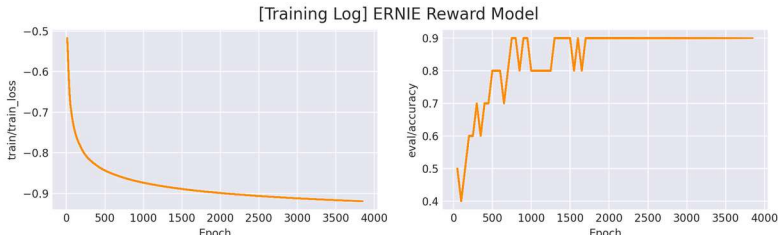
计算 rank_loss✦ 函数如下，因为样本里的句子已经默认按从高到低得分排好，因此我们只需要遍历的求前后项的得分差值加起来即可：

```
def compute_rank_list_loss(rank_rewards_list: List[List[torch.tensor]], device='cpu')
    """
    通过给定的有序（从高到低）的ranklist的reward列表，计算rank loss。
    所有排序高的句子的得分减去排序低的句子的得分差的总和，并取负。

    Args:
        rank_rewards_list (torch.tensor): 有序（从高到低）排序句子的reward列表，e.g. ->
                                          [
                                              [torch.tensor([0.3588]), torch.tensor([0.2
                                              [torch.tensor([0.5343]), torch.tensor([0.2
                                              ...
                                          ]
        device (str): 使用设备

    Returns:
        loss (torch.tensor): tensor([0.4891], grad_fn=<DivBackward0>)
    """
    if type(rank_rewards_list) != list:
        raise TypeError(f'@param rank_rewards expected "list", received {type(rank_rew

    loss, add_count = torch.tensor([0]).to(device), 0
    for rank_rewards in rank_rewards_list:
        for i in range(len(rank_rewards)-1):                                    # 遍历所
            for j in range(i+1, len(rank_rewards)):
                diff = F.sigmoid(rank_rewards[i] - rank_rewards[j])             # sigmo
                loss = loss + diff
                add_count += 1
    loss = loss / add_count
    return -loss
```

最终训练结果如下：

```
...
global step 10, epoch: 1, loss: -0.51766, speed: 0.21 step/s
global step 20, epoch: 1, loss: -0.55865, speed: 0.22 ste
global step 30, epoch: 1, loss: -0.60930, speed: 0.21 ste
```

```
global step 40, epoch: 1, loss: -0.65024, speed: 0.21 step/s
global step 50, epoch: 1, loss: -0.67781, speed: 0.22 step/s
Evaluation acc: 0.50000
best F1 performence has been updated: 0.00000 --> 0.50000
global step 60, epoch: 1, loss: -0.69296, speed: 0.20 step/s
global step 70, epoch: 1, loss: -0.70710, speed: 0.20 step/s
...
```



loss、acc 曲线图

我们输入两个评论句子：

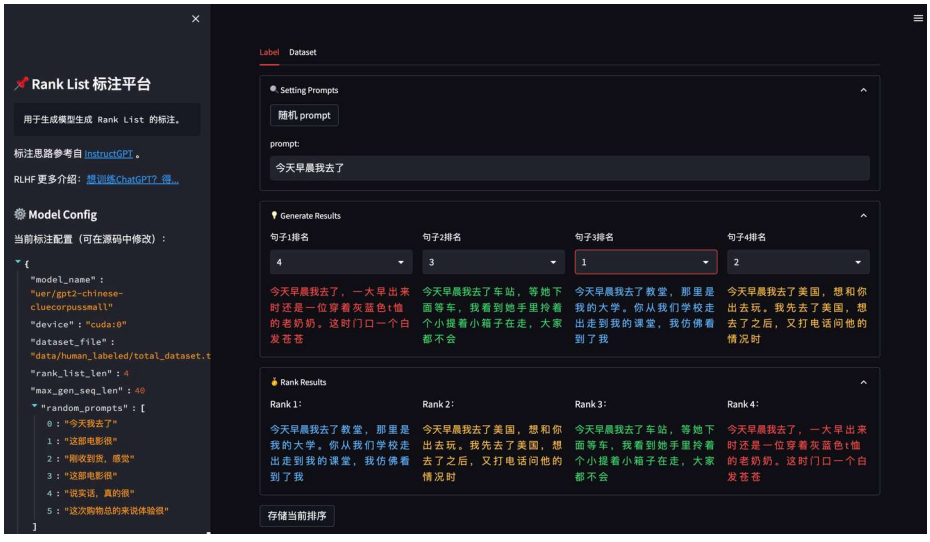
```
texts = [
 '买过很多箱这个苹果了，一如既往的好，汁多味甜～',
 '一台充电很慢，信号不好！退了！又买一台竟然是次品。。服了。。'
]

>>> tensor([[10.6989], [-9.2695]], grad_fn=<AddmmBackward>)
```

可以看到「正向评论」得到了 10.6 分，而「负向评论」得到了 -9.26 分。

4. 标注平台

在 InstructGPT 中是利用对语言模型（LM）的输出进行排序得到排序对从而训练 Reward Model。

如果想获得实现论文中类似的数据，在该项目中我们也提供了标注平台，可标注 rank_list 数据：



Rank List 标注平台（详情可参考源码仓库）

好啦，以上就是 Reward Model 的全部内容，感谢观看~

完整源码在这里：

https://github.com/HarderThenHarder/transformers_tasks/tree/main/RLHF
github.com/HarderThenHarder/transformers_

编辑于 2023-08-02 11:26 · IP 属地北京

内容所属专栏

深度学习（Deep Learning） ChatGPT 自然语言处理

61 条评论 默认 最新

何枝 作者
上一篇文章被和谐掉了，完整文章在这里：mp.weixin.qq.com/s/1v4U...
2023-05-05 · 北京 · 作者置顶 回复 2

强化学徒
就是不知道这个损失是他们的创新点还是早就有的技术
2022-12-31 · 安徽 回复 3

咸鱼二号
传统loss了吧，很多年前火的learning2rank也是这么搞的
2023-01-05 · 上海 回复 7

星宿海 ▸ **zhou reck**
是不是搞混了，这篇说的是用，你可能是在讲PPO里面RM怎样用，训RM那有什么Value Model。
2024-10-25 · 广东 回复 喜欢

展开其他 3 条回复 ›

mhcw
def compute_rank_list_loss(rank_rewards_list: List[List[torch.tensor]], device='cpu') -> torch.Tensor:
"""
通过给定的有序（从高到低）的ranklist的reward列表，计算rank loss。
所有排序高的句子的得分减去排序低的句子的得分差的总和，并取负。

Args:
rank_rewards_list (torch.tensor): 有序（从高到低）排序句子的reward列表，e.g. ->
[
[torch.tensor([0.3588]), torch.tensor([0.2481]), ...],
[torch.tensor([0.5343]), torch.tensor([0.2442]), ...],
...
]
device (str): 使用设备

```
Returns:
loss (torch.tensor): tensor([0.4891], grad_fn=<DivBackward0>)
"""
if type(rank_rewards_list) != list:
raise TypeError(f'@param rank_rewards expected "list", received
{type(rank_rewards)}.')

loss, add_count = torch.tensor([0]).to(device), 0
for rank_rewards in rank_rewards_list:
for i in range(len(rank_rewards)-1): # 遍历所有前项-后项的得分差
for j in range(i+1, len(rank_rewards)):
diff = F.logsigmoid(rank_rewards[i] - rank_rewards[j]) # sigmoid到0~1之间
loss = loss + diff
add_count += 1
loss = loss / add_count
return -loss
```

这个代码里好像没有取log的操作

2023-04-07 · 北京 回复 4

2333
确实
01-05 · 北京 回复 喜欢

杨柳
大佬 能讲解一下PPO算法么😘
2023-01-18 · 中国香港 回复 3

何枝 作者
记上ToDo😁
2023-01-19 · 北京 回复 2

conquer
请问一下prompt+response 排序如何制作数据集呢
2024-11-07 · 天津 回复 喜欢

Cpz已被占用
请问最后的损失-log(sigmoid(x))不应该大于0嘛，为什么图上的损失都是小于0呢
2024-08-20 · 北京 回复 喜欢

BreakableHeaven
最后reward score怎么算的?
2024-05-08 · 广东 回复 喜欢

妙脆角
您好，RM更新的时候应该不只是更新最后新加的线性层吧
2024-04-11 · 上海 回复 喜欢

wzz
真正的深入浅出，赞！
2024-04-01 · 浙江 回复 喜欢

点击查看全部评论 ›

理性发言，友善互动

译：感知用力比值（RPE）
2009-12-18 16:42:52| 分类： 体育
作者：Michael Tuchscherer 本文
摘自Michael Tuchscherer所著《反
应训练手册：制定适合你自己的力
量举训练计划》一书的
章谈的是反应训练（re

超等长练习的定义以及在训练中的运用

最新的
在外网d
分级表，
中英文对
Press：

safeguarddo

搞明白超等长练习之前我们要来说
挑两侧将几种收缩形式，分别是等
张收缩 等长收缩 等动收缩 超等长

力量举