

Day5 - 從nanoGPT開始 (4)

15th鐵人賽



jjchen1

團隊 我在鐵人賽烙賽、也在外木山裸泳owo

2023-09-06 23:37:20

373 瀏覽

昨天晚上睡覺的時間訓練了一版GPT用的是最小參數量的 **gpt2**，使用RTX3090訓練4000個iteration花了大約10個小時，我看設定好的MaxIteration=600,000，所以看來靠我的3090要訓練到模型收斂估計要62.5天左右；恩...不太意外，

```
config_args = {  
    'gpt2': dict(n_layer=12, n_head=12, n_embd=768), # 124M params  
    'gpt2-medium': dict(n_layer=24, n_head=16, n_embd=1024), # 350M params  
    'gpt2-large': dict(n_layer=36, n_head=20, n_embd=1280), # 774M params  
    'gpt2-xl': dict(n_layer=48, n_head=25, n_embd=1600), # 1558M params  
}
```

可以看到完整的Transformer結構非常的簡單，就是對我這種只有一張卡的人來說參數量大了一點 (124M)。

```
class GPT(nn.Module):  
    def __init__(self, config):  
        self.transformer = nn.ModuleDict(dict(  
            wte = nn.Embedding(config.vocab_size, config.n_embd),  
            wpe = nn.Embedding(config.block_size, config.n_embd),  
            drop = nn.Dropout(config.dropout),  
            h = nn.ModuleList([Block(config) for _ in range(config.n_layer)]),  
            ln_f = LayerNorm(config.n_embd, bias=config.bias),  
        ))  
        self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)  
        self.transformer.wte.weight = self.lm_head.weight # https://paperswithcode.com/method/weight-tying
```

model flops utilization (MFU)

我以前訓練網路其實沒在其他地方看到MFU的用法，在nanoGPT中會在每個iteration印出mfu來顯示顯卡的使用狀況，不過有一個問題是顯卡的能力是被寫死的 `flops_promised = 312e12 # A100 GPU bfloat16 peak flops is 312 TFLOPS`，像我的RTX3090的FP16算力是 35.58TFLOPS，所以必須要自己改一下，否則MFU的數值都會以A100為基準。

```
def estimate_mfu(self, fwdbwd_per_iter, dt):
    """ estimate model flops utilization (MFU) in units of A100 bfloat16 peak FLOPS """
    # first estimate the number of flops we do per iteration.
    # see PaLM paper Appendix B as ref: https://arxiv.org/abs/2204.02311
    N = self.get_num_params()
    cfg = self.config
    L, H, Q, T = cfg.n_layer, cfg.n_head, cfg.n_embd//cfg.n_head, cfg.block_size
    flops_per_token = 6*N + 12*L*H*Q*T
    flops_per_fwdbwd = flops_per_token * T
    flops_per_iter = flops_per_fwdbwd * fwdbwd_per_iter
    # express our flops throughput as ratio of A100 bfloat16 peak flops
    flops_achieved = flops_per_iter * (1.0/dt) # per second
    flops_promised = 312e12 # A100 GPU bfloat16 peak flops is 312 TFLOPS
    mfu = flops_achieved / flops_promised
    return mfu
```

看一看4000個iteration的nanoGPT的運算能力

- 訓練資料 `python data/openwebtext/prepare.py` 使用了huggingface上的 OpenWebText，由於事前處理訓練時所有iteration所需的資料，大約額外佔據硬碟130~150GB
- 訓練時使用預設的所有參數，直接執行 `python train.py`，預設大小為 gpt2，並且是不使用任何 pretrain

觀看訓練結果 `python sample.py`

- 我修改了檔案使用 `You are my sunshine.` 作為起始值，結果有點悲傷，不過也不意外，畢竟只有4000個iteration

You are my sunshine. Goodbye to the Wind. Bye, my, and thank you.

Edit: It is the only image I have ever taken.

Thanks to the Lobo team for the feedback!

Image: I have never uploaded anything.

...

由於結果有點爛無法接受，所以試試看下載作者訓練好的 `gpt2-xl` 看看，至少有語意的連貫了

- `python sample.py --init_from=gpt2-xl`

```
You are my sunshine. I will do anything for you."

"The Sun is the Sun. Nothing else matters. You are the Sun."

>Your feelings are the sun."

"You are my sunshine. I will do anything for you."

"The Sun is the Sun. Nothing else matters. You are the Sun."
....
```

不過很常碰到一個問題就是GPT的生成內容會碰到某些句子的時候就會卡住，並且開始無限重複，很久以前就聽說過GPT-2的模型有這個問題，不過今天又在次見到了，類似這樣的問題，頻繁發生。

```
...
I am a writer.

I am a human writer.

I am a human writer.

I am a human writer.

I am a philosopher.

I am a poet.

I am an author.

I am a writer.

I am a writer.
...
```

這個問題我問了一下ChatGPT，我覺得他的回答很不錯，所以就借花獻佛貼上來一下：

問

我記得GPT2剛出來的時候，很容易碰到一個問題就是，開始生成某些句子之後，那些句子後面就很容易造成重複的結果，最後GPT就不斷的repeat同一個句子；後面的研究是如何避免這個問題？造成這個問題的原因是什麼？

答

GPT-2在剛推出時確實有一個問題，被稱為"重複性問題"或"循環性問題" (**Repetition or Looping Problem**)，這是因為模型的生成過程是基於概率的，而且它在生成文本時是一次一個詞彙地進行的。當模型生成一個詞彙時，它只能看到先前生成的詞彙，並根據它所見的內容預測下一個詞彙。這種方式容易導致模型陷入循環，生成重複的內容。

造成這個問題的原因主要有幾個：

缺乏全局語境：GPT-2是一個無記憶的模型，它無法記住已經生成的內容，因此無法在全局上理解整個文本的語境。它僅依賴於先前生成的詞彙來預測下一個詞彙，這可能導致生成的文本失去多樣性。

高機率詞彙：某些詞彙可能在給定語境下具有高概率，因此模型更容易選擇這些詞彙，導致重複性。例如，對於一個以"狗"結尾的句子，"狗"後面很可能是"是"或"很可愛"，這些詞彙的概率較高。

後來的研究通過以下方式來解決這個問題：

集束搜索 (**Beam Search**)：集束搜索是一種生成文本的方法，它考慮多個可能的詞彙序列，而不僅僅是單個最可能的詞彙。這樣可以增加文本的多樣性，減少重複性。

溫度參數 (**Temperature Parameter**)：溫度參數可以調整模型生成文本的隨機性。較高的溫度會使生成更加隨機，較低的溫度會使生成更加確定。調整溫度可以控制文本的多樣性，減少重複性。

刪除重複：在生成的文本中，可以檢測和刪除重複的片段，從而減少內容的循環性。

長文本處理：對於生成長文本，可以引入更多的上下文信息，並使用技術如分段生成 (**Chunking**) 來幫助模型更好地理解全局語境。

這些方法結合使用可以改善GPT-2和類似模型的生成質量，減少重複性問題，並使生成的文本更具多樣性和連貫性。然而，模型的改進還在繼續進行，未來的研究可能會提供更多解決這個問題的方法。

本來打算自己使用nanoGPT訓練能夠有加減乘除計算能力的GPT，不過做這個實驗需要不少時間，一天也就能擠出一個小時的時間來做，因此還沒做完，可能要先欠著等以後補上；nanoGPT的程式碼都有很詳細的註解，以及提供了很多用法的參考網址，或至少都會附上對應的專有名詞以供查找，非常

具有學習價值，打算做實驗的時候慢慢深度的探索。

明天打算開始前往baby llama2專案，見識一下一千行以內的c語言實做出的llama2。