Deep HW4 – Part 3 report

-1

مدل ها را ایمپورت میکنیم

-2

به ازای سه ورودی دلخواه ادامه جملات را generate میکنیم.

Input: I think it will

Output: I think it will be a good idea to keep it in the library for future use," said Michael L. Shulkin, executive director of the Institute for Advanced Study (IAS).

Input: The movie was

Output: The movie was released in English, but the English version has a few minor errors. In one scene one character is holding her hand and another is holding the other's back. The second scene depicts a man wearing a T-shirt and holding his head

Input: The product is

Output: The product is ready to go to the customer's house.

-3

پس از لود کردن دیتاست با استفاده از از توکنایزر و دستور زیر padding و truncation انجام میدهیم تا سایز ورودی ها به 128 برسد.

```
tokenizer(examples["sentence"], padding="max_length", truncation=True, max_length=MAX_LEN)
```

4 و 5-

در این بخش لایه آخر gpt2 را به تابع همانی تغییر داده و پس از فریز کردن مدل از CLS token ها برای طبقه بندی استفاده میکنیم. یکبار فقط با استفاده از آخرین توکن و بار دیگر با استفاده از تمام وکن ها. در هر دو حالت دقت مدل کمتر از 60 درصد است.

```python
class GPT2WithLinearClassifier(torch.nn.Module):
    def __init__(self, gpt2_model):
        super(GPT2WithLinearClassifier, self).__init__()
        self.gpt2 = gpt2_model

        last_layer = nn.Linear(self.gpt2.config.hidden_size, self.gpt2.config.hidden_size, bias=True)

        with torch.no_grad():
          last_layer.weight.copy_(torch.eye(self.gpt2.config.hidden_size))  # Identity matrix
          last_layer.bias.zero_()  # Zero bias

        self.gpt2.lm_head = last_layer

        for param in self.gpt2.parameters():
            param.reguires_grad = False

        self.linear = torch.nn.Linear(self.gpt2.config.hidden_size, 1)
        self.activation = nn.Sigmoid()

        print(self.gpt2.config.hidden_size)

    def forward(self, input_ids, attention_mask=None):
        output = self.gpt2(input_ids, attention_mask=attention_mask)
        cls_embedding = output['logits'][:, 0, :]

        return torch.squeeze(self.activation(self.linear(cls_embedding)))
```

```python
class GPT2_Linear(torch.nn.Module):
    def __init__(self, gpt2_model):
        super(GPT2_Linear, self).__init__()
        self.gpt2 = gpt2_model

        last_layer = nn.Linear(self.gpt2.config.hidden_size, self.gpt2.config.hidden_size, bias=True)

        with torch.no_grad():
          last_layer.weight.copy_(torch.eye(self.gpt2.config.hidden_size))  # Identity matrix
          last_layer.bias.zero_()  # Zero bias

        self.gpt2.lm_head = last_layer

        for param in self.gpt2.parameters():
            param.reguires_grad = False

        self.linear = nn.Sequential(
                    nn.Flatten(),
                    nn.Linear(self.gpt2.config.hidden_size*128, 128),
                    nn.ReLU(),
                    nn.Linear(128, 1),
                    nn.Sigmoid()
        )


        self.activation = nn.Sigmoid()

    def forward(self, input_ids, attention_mask=None):
        output = self.gpt2(input_ids, attention_mask=attention_mask)
        cls_embedding = output['logits']

        return torch.squeeze(self.linear(cls_embedding))
```

-6

در این بخش CLS token ها را به یک مدول attention میدهیم و سپس از یک شبکه fully connected عبور میدهیم تا طبقه بندی به خوبی انجام شود.

```python
class BidirectionalAttentionClassifier(torch.nn.Module):
    def __init__(self, gpt2_model, num_heads=12):
        super().__init__()
        self.gpt2 = gpt2_model

        for param in self.gpt2.parameters():
          param.requires_grad = False

        self.multihead_attn = torch.nn.MultiheadAttention(self.gpt2.config.hidden_size, num_heads
        self.linear = nn.Sequential(
                    nn.Flatten(),
                    nn.Linear(768, 128),
                    nn.ReLU(),
                    nn.Linear(128, 128),
                    nn.ReLU(),
                    nn.Linear(128, 128),
                    nn.ReLU(),
                    nn.Linear(128, 1),
                    nn.Sigmoid()
        )


    def forward(self, input_ids, attention_mask=None):
        output = self.gpt2(input_ids, attention_mask=attention_mask)

        attn_output, _ = self.multihead_attn(output.last_hidden_state[:, 0, :].unsqueeze(0),
                                             output.last_hidden_state.transpose(0, 1),
                                             output.last_hidden_state.transpose(0, 1))

        return torch.squeeze(self.linear(attn_output.squeeze(0)))
```

مدل را به اندازه 2 ایپاک آموزش میدهیم و به دقت 84.29% بر روی داده های ارزیابی میرسیم.

```
100%|          | 4210/4210 [14:59<00:00,  4.68batch/s, loss=0.509, metric=0.74]
Model Saved!
Valid: Loss = 0.381, Metric = 0.8268

Epoch 1: 100%|          | 4210/4210 [14:58<00:00,  4.68batch/s, loss=0.437, metric=0.798]
Model Saved!
Valid: Loss = 0.3756, Metric = 0.8429
```