10.6 Summary

In this chapter, I have introduced the topic of Transformers. I discussed the main ideas used in Transformers and described in detail the algorithm for a GPT.

11 Transfer Learning

Transfer Learning is one of the most important developments in AI since Transformers. Transfer Learning is an approach where you pre-train a model with some data (usually a lot of data) and then fine-tune the same model with less data but for a specific task. This allows companies (entities) with vast resources to pre-train models that they can later share with others. Training with massively large data sets requires a lot of resources such as computing power, money, and can currently only be done by large institutions and companies such as Google, Meta, etc.

11.0.1 Important Ideas in Transfer Learning

Some of the most important ideas in Transfer Learning include:

- Pre-training
- Fine-tuning
- Freezing some weights of the neural network
- Sharing the weights from one model to another
- Reinforcement Learning Through Human Feedbacks (RLHF)
- Transfer Learning Metrics

11.0.2 Pre-training

Pre-training is the process of training a model like a Large Language Model (LLM) on massive amounts of text data. In the past, the best models have been the ones pre-trained by large companies such as OpenAI, Google, and Meta. Pre-training starts with a model that has had no training and where the weights are usually initialized randomly. The process can take many months, require multiple GPUs, and training needs to be performed on distributed GPU systems.

You start with code such as that of the GPT algorithm but you need to train the model for long periods of time with lots of data.

11.0.3 Fine-tuning

Fine-tuning is the process of taking a pre-trained model with pre-trained weights and performing some additional training on it. Here, you want to re-use as much of the pre-trained weights as possible. One scheme for this requires freezing some layers of the network so that those corresponding weights are not changed during fine-tuning. Fine tuning can still

be done on one GPU, currently, and may not require massive amounts of data. It is usually focused on specific tasks like sentiment analysis, question answering, etc.

11.0.4 Freezing some weights of the neural network

When performing fine-tuning, freezing part of the weights of the model may be required. This process can be seen in the next code listing.

Listing 11.1: Freezing the weights

```
## Freeze the first 70% of the hidden layers of the model

layers              = model_gpt.transformer.h
num_layers          = len(layers)
print( num_layers )

## 12

num_unfrozen        = int(0.3 * num_layers)

for layer in layers[:-num_unfrozen]:
    layer.requires_grad_(False)

## check which frozen

for name, param in model_gpt_gen_reward.named_parameters():
     print(name, param.requires_grad)
```

Notice that in the previous code listing we iterate through all the layers and change the parameter **layer.requires_grad_()** to **False** for all the layers we want to freeze. The result of freezing can be seen below for a GPT2.

Listing 11.2: Result of Freezing the weights

```
transformer.wte.weight True
transformer.wpe.weight True
transformer.h.0.ln_1.weight False
transformer.h.0.ln_1.bias False
transformer.h.0.attn.c_attn.weight False
transformer.h.0.attn.c_attn.bias False
transformer.h.0.attn.c_proj.weight False
transformer.h.0.attn.c_proj.bias False
transformer.h.0.ln_2.weight False
transformer.h.0.ln_2.bias False
...
transformer.h.8.ln_1.weight False
transformer.h.8.ln_1.bias False
transformer.h.8.attn.c_attn.weight False
transformer.h.8.attn.c_attn.bias False
transformer.h.8.attn.c_proj.weight False
transformer.h.8.attn.c_proj.bias False
transformer.h.8.ln_2.weight False
transformer.h.8.ln_2.bias False
transformer.h.8.mlp.c_fc.weight False
transformer.h.8.mlp.c_fc.bias False
transformer.h.8.mlp.c_proj.weight False
transformer.h.8.mlp.c_proj.bias False
transformer.h.9.ln_1.weight True
transformer.h.9.ln_1.bias True
transformer.h.9.attn.c_attn.weight True
transformer.h.9.attn.c_attn.bias True
transformer.h.9.attn.c_proj.weight True
transformer.h.9.attn.c_proj.bias True
transformer.h.9.ln_2.weight True
transformer.h.9.ln_2.bias True
transformer.h.9.mlp.c_fc.weight True
transformer.h.9.mlp.c_fc.bias True
transformer.h.9.mlp.c_proj.weight True
transformer.h.9.mlp.c_proj.bias True
...
transformer.h.11.ln_1.weight True
transformer.h.11.ln_1.bias True
transformer.h.11.attn.c_attn.weight True
transformer.h.11.attn.c_attn.bias True
transformer.h.11.attn.c_proj.weight True
transformer.h.11.attn.c_proj.bias True
transformer.h.11.ln_2.weight True
transformer.h.11.ln_2.bias True
transformer.h.11.mlp.c_fc.weight True
transformer.h.11.mlp.c_fc.bias True
transformer.h.11.mlp.c_proj.weight True
transformer.h.11.mlp.c_proj.bias True
transformer.ln_f.weight True
transformer.ln_f.bias True
v_head.weight True
```

11.0.5 Sharing the weights from one model to another

Sharing weights is very important as it allows pre-training to be done only once. After that, only some modification to the weights needs to be done. In torch, the weights can be extracted easily using, for example, the function **model.parameters()**.

Currently, there are specialized libraries that help share models. These include HuggingFace's Transformers module and Fastai.

11.0.6 Reinforcement Learning Through Human Feedbacks (RLHF)

RLHF is a technique from RL were you train a model by providing feed backs. Since manually creating a reward function to create feed backs is very difficult, this can be done by replacing the reward function with another neural network.

For example, we can train a GPT model to generate more positive or more negative movie reviews. The process involves giving text prompts about movies to the GPT in order for it to generate text responses from those prompts. These generated text responses (movie reviews) and the original prompts are concatenated together and given to a reward function for evaluation. The reward function will assign a score to the input text indicating how good it is. The reward could be based on heuristic rules, human feedbacks, etc. A clever approach is to use a task specific BERT model to read in the GPT generated text and assign a score (reward) given this text. As such, the BERT model can be used as the reward function.

11.0.7 Transfer Learning Performance Metrics

Since Fine-tuned models are highly specialized models that can perform special tasks such as translation, or summarization, they may require new special metrics for performance analysis. Some common metrics for this can be seen in the list below.

- BLEU
- WER
- Glue
- etc.

A great survey paper with many of these type of metrics can be found here: Celikyilmaz, Clark, and Gao, 2021.

11.1 Summary

This chapter presented some aspects of Transfer Learning. Some of the most important ideas were presented and discussed.