```
hprams.json: 100%|████████████████████████| 90.0/90.0 [00:00<00:00,
                                                    78.3kiB/s]
model.ckpt.data-00000-of-00001: 100%|███████| 498M/498M [01:09<00:00,
                                                    7.16MiB/s]
model.ckpt.index: 100%|█████████████████████| 5.21k/5.21k [00:00<00:00,
                                                      3.24MiB/s]
model.ckpt.meta: 100%|██████████████████████| 471k/471k [00:00<00:00,
                                                    2.46MiB/s]
vocab.bpe: 100%|████████████████████████████| 456k/456k [00:00<00:00,
                                                    1.70MiB/s]
```

> **NOTE** If the download code does not work for you, it could be due to intermittent internet connection, server problems, or changes in how OpenAI shares the weights of the open-source GPT-2 model. In this case, please visit this chapter's online code repository at https://github.com/rasbt/LLMs-from-scratch for alternative and updated instructions, and reach out via the Manning Forum for further questions.

Assuming the execution of the previous code has completed, let's inspect the contents of `settings` and `params`:

```
print("Settings:", settings)
print("Parameter dictionary keys:", params.keys())
```

The contents are

```
Settings: {'n_vocab': 50257, 'n_ctx': 1024, 'n_embd': 768, 'n_head': 12,
           'n_layer': 12}
Parameter dictionary keys: dict_keys(['blocks', 'b', 'g', 'wpe', 'wte'])
```

Both `settings` and `params` are Python dictionaries. The `settings` dictionary stores the LLM architecture settings similarly to our manually defined `GPT_CONFIG_124M` settings. The `params` dictionary contains the actual weight tensors. Note that we only printed the dictionary keys because printing the weight contents would take up too much screen space; however, we can inspect these weight tensors by printing the whole dictionary via `print(params)` or by selecting individual tensors via the respective dictionary keys, for example, the embedding layer weights:

```
print(params["wte"])
print("Token embedding weight tensor dimensions:", params["wte"].shape)
```

The weights of the token embedding layer are

```
[[-0.11010301 ... -0.1363697   0.01506208   0.04531523]
 [ 0.04034033 ...  0.08605453  0.00253983   0.04318958]
 [-0.12746179  ...  0.08991534 -0.12972379 -0.08785918]
 ...
 [-0.04453601 ...   0.10435229  0.09783269 -0.06952604]
 [ 0.1860082  ...  -0.09625227  0.07847701 -0.02245961]
```

```
 [ 0.05135201 ...   0.00704835  0.15519823  0.12067825]]
Token embedding weight tensor dimensions: (50257, 768)
```

We downloaded and loaded the weights of the smallest GPT-2 model via the `download_and_load_gpt2(model_size="124M", ...)` setting. OpenAI also shares the weights of larger models: `355M`, `774M`, and `1558M`. The overall architecture of these differently sized GPT models is the same, as illustrated in figure 5.17, except that different

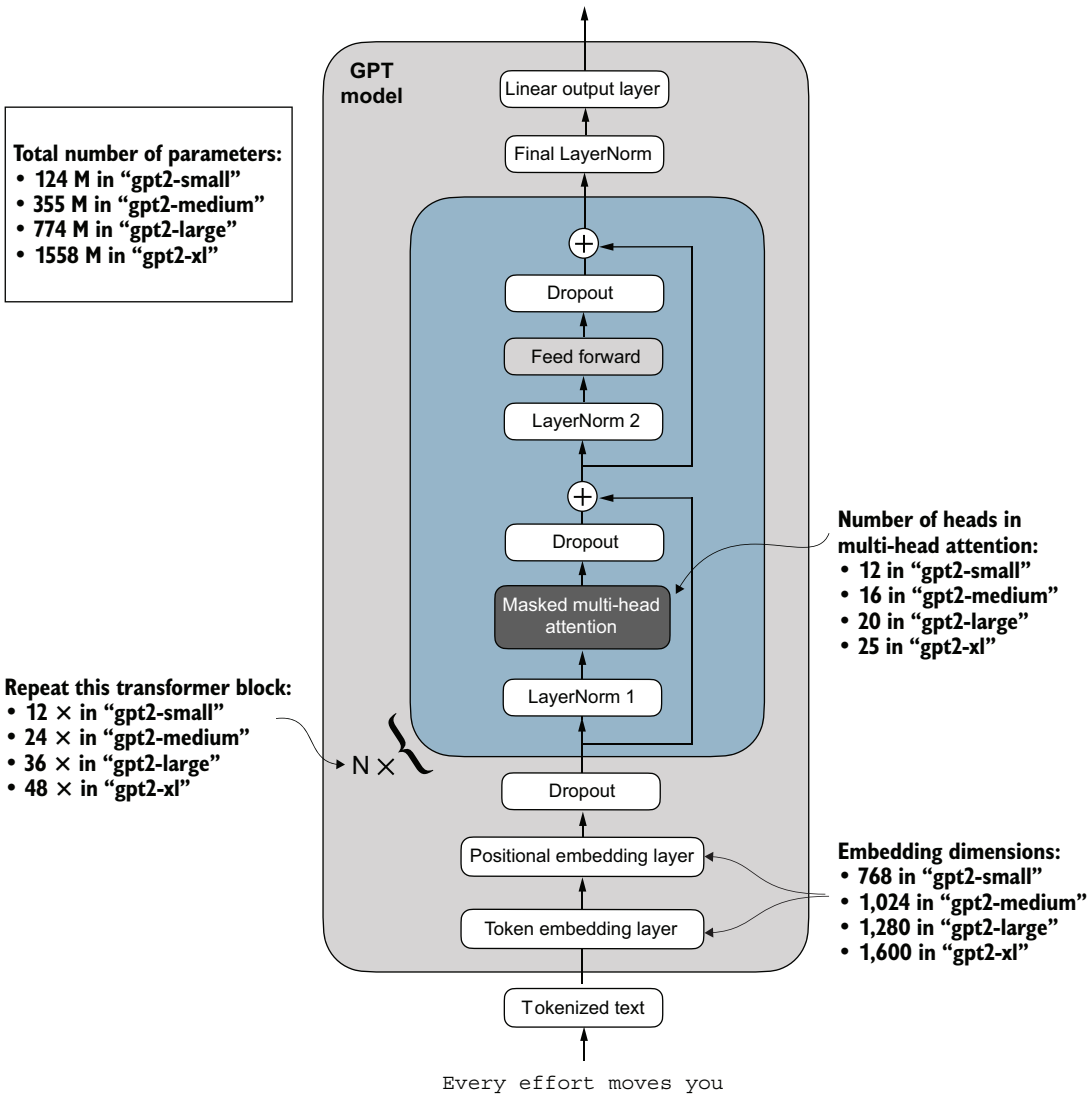

Figure 5.17 GPT-2 LLMs come in several different model sizes, ranging from 124 million to 1,558 million parameters. The core architecture is the same, with the only difference being the embedding sizes and the number of times individual components like the attention heads and transformer blocks are repeated.

architectural elements are repeated different numbers of times and the embedding size differs. The remaining code in this chapter is also compatible with these larger models.

After loading the GPT-2 model weights into Python, we still need to transfer them from the `settings` and `params` dictionaries into our `GPTModel` instance. First, we create a dictionary that lists the differences between the different GPT model sizes in figure 5.17:

```
model_configs = {
    "gpt2-small (124M)": {"emb_dim": 768, "n_layers": 12, "n_heads": 12},
    "gpt2-medium (355M)": {"emb_dim": 1024, "n_layers": 24, "n_heads": 16},
    "gpt2-large (774M)": {"emb_dim": 1280, "n_layers": 36, "n_heads": 20},
    "gpt2-xl (1558M)": {"emb_dim": 1600, "n_layers": 48, "n_heads": 25},
}
```

Suppose we are interested in loading the smallest model, `"gpt2-small (124M)"`. We can use the corresponding settings from the `model_configs` table to update our full-length `GPT_CONFIG_124M` we defined and used earlier:

```
model_name = "gpt2-small (124M)"
NEW_CONFIG = GPT_CONFIG_124M.copy()
NEW_CONFIG.update(model_configs[model_name])
```

Careful readers may remember that we used a 256-token length earlier, but the original GPT-2 models from OpenAI were trained with a 1,024-token length, so we have to update the `NEW_CONFIG` accordingly:

```
NEW_CONFIG.update({"context_length": 1024})
```

Also, OpenAI used bias vectors in the multi-head attention module's linear layers to implement the query, key, and value matrix computations. Bias vectors are not commonly used in LLMs anymore as they don't improve the modeling performance and are thus unnecessary. However, since we are working with pretrained weights, we need to match the settings for consistency and enable these bias vectors:

```
NEW_CONFIG.update({"qkv_bias": True})
```

We can now use the updated `NEW_CONFIG` dictionary to initialize a new `GPTModel` instance:

```
gpt = GPTModel(NEW_CONFIG)
gpt.eval()
```