Processing the dataset takes about 1 minute on an A100 GPU and 6 minutes on an M3 MacBook Air:

```
100%|██████████| 110/110 [01:05<00:00,  1.68it/s]
```

Let's verify that the responses have been correctly added to the `test_set` dictionary by examining one of the entries:

```
print(test_data[0])
```

The output shows that the `model_response` has been added correctly:

```
{'instruction': 'Rewrite the sentence using a simile.',
 'input': 'The car is very fast.',
 'output': 'The car is as fast as lightning.',
 'model_response': 'The car is as fast as a bullet.'}
```

Finally, we save the model as `gpt2-medium355M-sft.pth` file to be able to reuse it in future projects:

```
import re

file_name = f"{re.sub(r'[ ()]', '', CHOOSE_MODEL) }-sft.pth"
torch.save(model.state_dict(), file_name)
print(f"Model saved as {file_name}")
```

Removes white spaces and parentheses from file name

The saved model can then be loaded via `model.load_state_dict(torch.load("gpt2-medium355M-sft.pth"))`.

7.8 Evaluating the fine-tuned LLM

Previously, we judged the performance of an instruction-fine-tuned model by looking at its responses on three examples of the test set. While this gives us a rough idea of how well the model performs, this method does not scale well to larger amounts of responses. So, we implement a method to automate the response evaluation of the fine-tuned LLM using another, larger LLM, as highlighted in figure 7.19.

To evaluate test set responses in an automated fashion, we utilize an existing instruction-fine-tuned 8-billion-parameter Llama 3 model developed by Meta AI. This model can be run locally using the open source Ollama application (https://ollama.com).

> **NOTE** Ollama is an efficient application for running LLMs on a laptop. It serves as a wrapper around the open source llama.cpp library (https://github.com/ggerganov/llama.cpp), which implements LLMs in pure C/C++ to maximize efficiency. However, Ollama is only a tool for generating text using LLMs (inference) and does not support training or fine-tuning LLMs.

Licensed to Jason Walsh <manning@jwalsh.net>

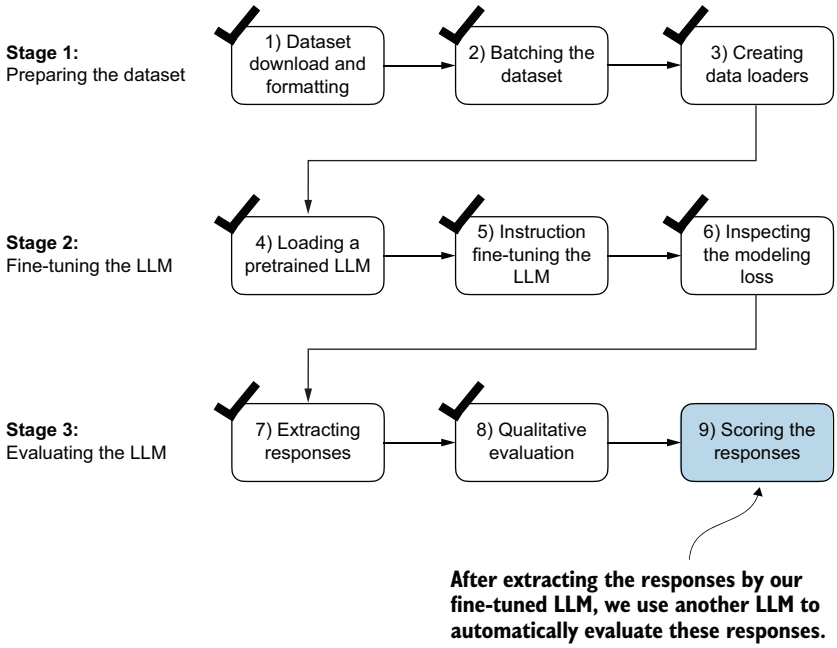

Figure 7.19 The three-stage process for instruction fine-tuning the LLM. In this last step of the instruction-fine-tuning pipeline, we implement a method to quantify the performance of the fine-tuned model by scoring the responses it generated for the test.

Using larger LLMs via web APIs

The 8-billion-parameter Llama 3 model is a very capable LLM that runs locally. However, it's not as capable as large proprietary LLMs such as GPT-4 offered by OpenAI. For readers interested in exploring how to utilize GPT-4 through the OpenAI API to assess generated model responses, an optional code notebook is available within the supplementary materials accompanying this book at https://mng.bz/BgEv.

To execute the following code, install Ollama by visiting https://ollama.com and follow the provided instructions for your operating system:

- *For macOS and Windows users*—Open the downloaded Ollama application. If prompted to install command-line usage, select Yes.
- *For Linux users*—Use the installation command available on the Ollama website.

Before implementing the model evaluation code, let's first download the Llama 3 model and verify that Ollama is functioning correctly by using it from the command-line terminal. To use Ollama from the command line, you must either start the Ollama application or run `ollama serve` in a separate terminal, as shown in figure 7.20.

Licensed to Jason Walsh <manning@jwalsh.net>

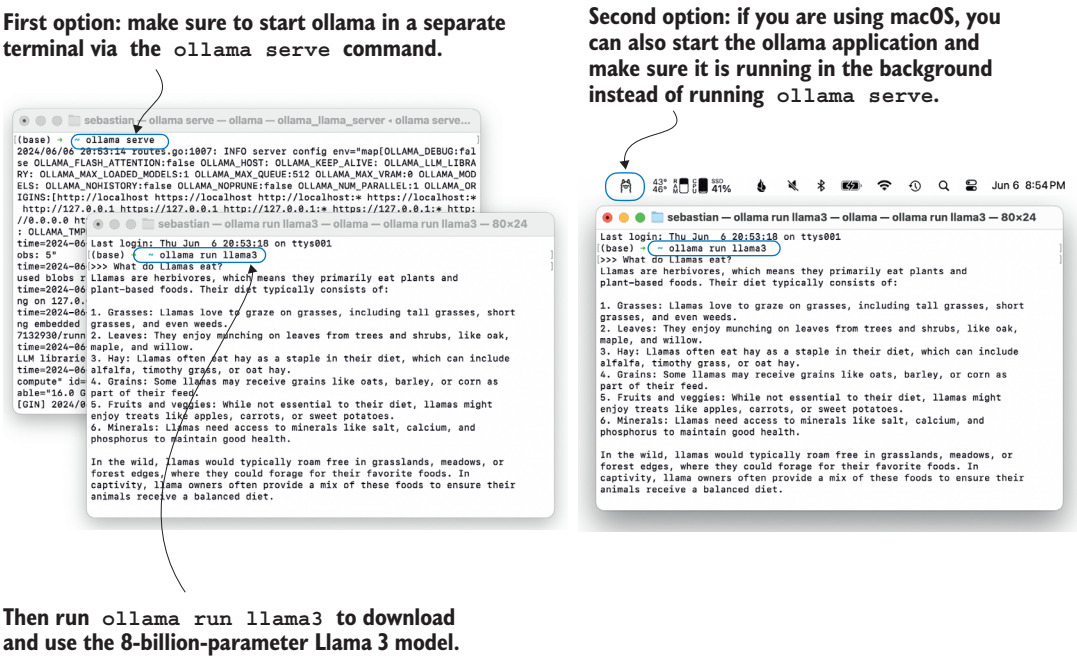

Figure 7.20 Two options for running Ollama. The left panel illustrates starting Ollama using `ollama serve`. The right panel shows a second option in macOS, running the Ollama application in the background instead of using the `ollama serve` command to start the application.

With the Ollama application or `ollama serve` running in a different terminal, execute the following command on the command line (not in a Python session) to try out the 8-billion-parameter Llama 3 model:

```
ollama run llama3
```

The first time you execute this command, this model, which takes up 4.7 GB of storage space, will be automatically downloaded. The output looks like the following:

```
pulling manifest
pulling 6a0746a1ec1a... 100% |████████████████| 4.7 GB
pulling 4fa551d4f938... 100% |████████████████|  12 KB
pulling 8ab4849b038c... 100% |████████████████|  254 B
pulling 577073ffcc6c... 100% |████████████████|  110 B
pulling 3f8eb4da87fa... 100% |████████████████|  485 B
verifying sha256 digest
writing manifest
removing any unused layers
success
```

Licensed to Jason Walsh <manning@jwalsh.net>