

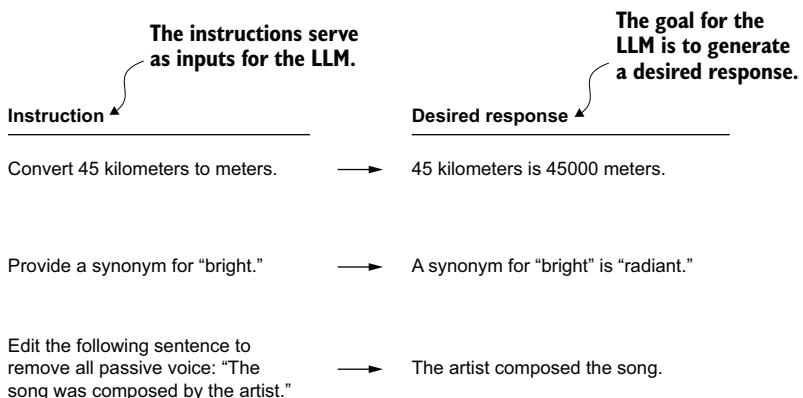
Figure 7.1 The three main stages of coding an LLM. This chapter focuses on step 9 of stage 3: fine-tuning a pretrained LLM to follow human instructions.

Figure 7.1 shows two main ways of fine-tuning an LLM: fine-tuning for classification (step 8) and fine-tuning an LLM to follow instructions (step 9). We implemented step 8 in chapter 6. Now we will fine-tune an LLM using an *instruction dataset*.

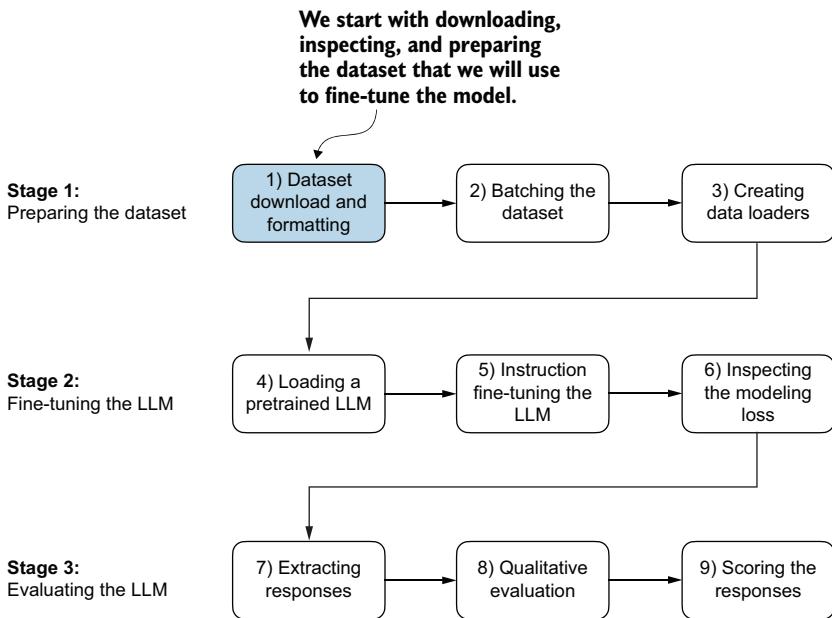
## 7.1 Introduction to instruction fine-tuning

We now know that pretraining an LLM involves a training procedure where it learns to generate one word at a time. The resulting pretrained LLM is capable of *text completion*, meaning it can finish sentences or write text paragraphs given a fragment as input. However, pretrained LLMs often struggle with specific instructions, such as “Fix the grammar in this text” or “Convert this text into passive voice.” Later, we will examine a concrete example where we load the pretrained LLM as the basis for *instruction fine-tuning*, also known as *supervised instruction fine-tuning*.

Here, we focus on improving the LLM’s ability to follow such instructions and generate a desired response, as illustrated in figure 7.2. Preparing the dataset is a key aspect of instruction fine-tuning. Then we’ll complete all the steps in the three stages of the instruction fine-tuning process, beginning with the dataset preparation, as shown in figure 7.3.



**Figure 7.2 Examples of instructions that are processed by an LLM to generate desired responses**



**Figure 7.3 The three-stage process for instruction fine-tuning an LLM. Stage 1 involves dataset preparation, stage 2 focuses on model setup and fine-tuning, and stage 3 covers the evaluation of the model. We will begin with step 1 of stage 1: downloading and formatting the dataset.**

## 7.2 Preparing a dataset for supervised instruction fine-tuning

Let's download and format the instruction dataset for instruction fine-tuning a pre-trained LLM. The dataset consists of 1,100 *instruction–response pairs* similar to those in figure 7.2. This dataset was created specifically for this book, but interested readers can find alternative, publicly available instruction datasets in appendix B.

The following code implements and executes a function to download this dataset, which is a relatively small file (only 204 KB) in JSON format. JSON, or JavaScript Object Notation, mirrors the structure of Python dictionaries, providing a simple structure for data interchange that is both human readable and machine friendly.

**Listing 7.1 Downloading the dataset**

```
import json
import os
import urllib

def download_and_load_file(file_path, url):
    if not os.path.exists(file_path):
        with urllib.request.urlopen(url) as response:
            text_data = response.read().decode("utf-8")
        with open(file_path, "w", encoding="utf-8") as file:
            file.write(text_data)
    with open(file_path, "r") as file:
        data = json.load(file)
    return data

file_path = "instruction-data.json"
url = (
    "https://raw.githubusercontent.com/rasbt/LLMs-from-scratch"
    "/main/ch07/01_main-chapter-code/instruction-data.json"
)

data = download_and_load_file(file_path, url)
print("Number of entries:", len(data))
```

The output of executing the preceding code is

```
Number of entries: 1100
```

The `data` list that we loaded from the JSON file contains the 1,100 entries of the instruction dataset. Let's print one of the entries to see how each entry is structured:

```
print("Example entry:\n", data[50])
```