or for experimenting with different training configurations to improve the model's performance.

It's worth noting that Ollama is not entirely deterministic across operating systems at the time of this writing, which means that the scores you obtain might vary slightly from the previous scores. To obtain more robust results, you can repeat the evaluation multiple times and average the resulting scores.

To further improve our model's performance, we can explore various strategies, such as

- Adjusting the hyperparameters during fine-tuning, such as the learning rate, batch size, or number of epochs
- Increasing the size of the training dataset or diversifying the examples to cover a broader range of topics and styles
- Experimenting with different prompts or instruction formats to guide the model's responses more effectively
- Using a larger pretrained model, which may have greater capacity to capture complex patterns and generate more accurate responses

NOTE For reference, when using the methodology described herein, the Llama 3 8B base model, without any fine-tuning, achieves an average score of 58.51 on the test set. The Llama 3 8B instruct model, which has been fine-tuned on a general instruction-following dataset, achieves an impressive average score of 82.6.

Exercise 7.4 Parameter-efficient fine-tuning with LoRA

To instruction fine-tune an LLM more efficiently, modify the code in this chapter to use the low-rank adaptation method (LoRA) from appendix E. Compare the training run time and model performance before and after the modification.

7.9 Conclusions

This chapter marks the conclusion of our journey through the LLM development cycle. We have covered all the essential steps, including implementing an LLM architecture, pretraining an LLM, and fine-tuning it for specific tasks, as summarized in figure 7.21. Let's discuss some ideas for what to look into next.

7.9.1 What's next?

While we covered the most essential steps, there is an optional step that can be performed after instruction fine-tuning: preference fine-tuning. Preference fine-tuning is particularly useful for customizing a model to better align with specific user preferences. If you are interested in exploring this further, see the `04_preference-tuning-with-dpo` folder in this book's supplementary GitHub repository at https://mng.bz/dZwD.

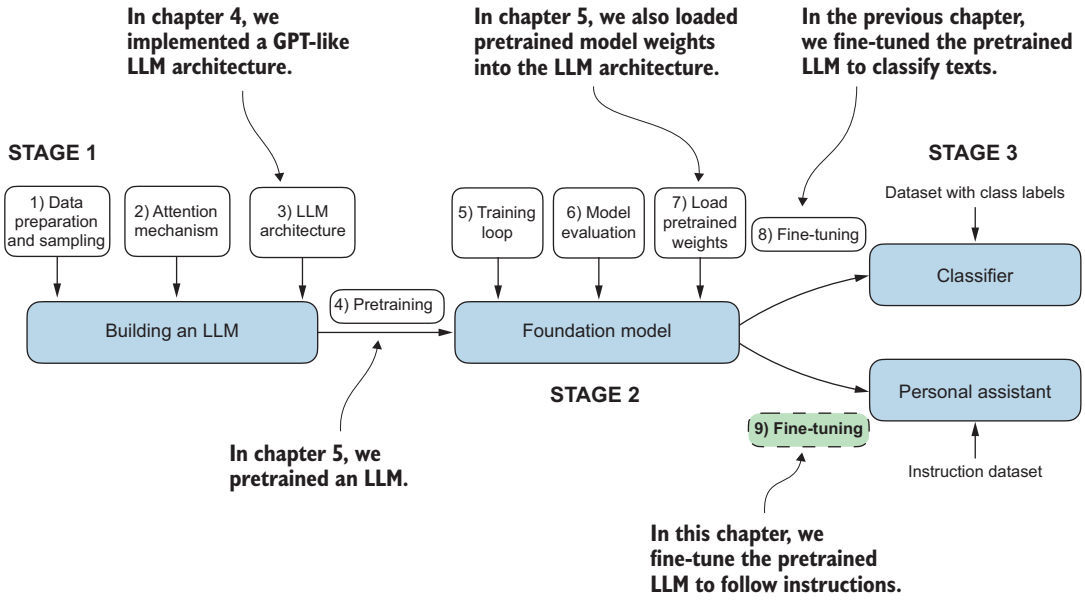

Figure 7.21 The three main stages of coding an LLM.

In addition to the main content covered in this book, the GitHub repository also contains a large selection of bonus material that you may find valuable. To learn more about these additional resources, visit the Bonus Material section on the repository's README page: https://mng.bz/r12g.

7.9.2 Staying up to date in a fast-moving field

The fields of AI and LLM research are evolving at a rapid (and, depending on who you ask, exciting) pace. One way to keep up with the latest advancements is to explore recent research papers on arXiv at https://arxiv.org/list/cs.LG/recent. Additionally, many researchers and practitioners are very active in sharing and discussing the latest developments on social media platforms like X (formerly Twitter) and Reddit. The subreddit r/LocalLLaMA, in particular, is a good resource for connecting with the community and staying informed about the latest tools and trends. I also regularly share insights and write about the latest in LLM research on my blog, available at https://magazine.sebastianraschka.com and https://sebastianraschka.com/blog/.

7.9.3 Final words

I hope you have enjoyed this journey of implementing an LLM from the ground up and coding the pretraining and fine-tuning functions from scratch. In my opinion, building an LLM from scratch is the most effective way to gain a deep understanding of how LLMs work. I hope that this hands-on approach has provided you with valuable insights and a solid foundation in LLM development.

While the primary purpose of this book is educational, you may be interested in utilizing different and more powerful LLMs for real-world applications. For this, I recommend exploring popular tools such as Axolotl (https://github.com/OpenAccess-AI-Collective/axolotl) or LitGPT (https://github.com/Lightning-AI/litgpt), which I am actively involved in developing.

Thank you for joining me on this learning journey, and I wish you all the best in your future endeavors in the exciting field of LLMs and AI!

Summary

- The instruction-fine-tuning process adapts a pretrained LLM to follow human instructions and generate desired responses.
- Preparing the dataset involves downloading an instruction-response dataset, formatting the entries, and splitting it into train, validation, and test sets.
- Training batches are constructed using a custom collate function that pads sequences, creates target token IDs, and masks padding tokens.
- We load a pretrained GPT-2 medium model with 355 million parameters to serve as the starting point for instruction fine-tuning.
- The pretrained model is fine-tuned on the instruction dataset using a training loop similar to pretraining.
- Evaluation involves extracting model responses on a test set and scoring them (for example, using another LLM).
- The Ollama application with an 8-billion-parameter Llama model can be used to automatically score the fine-tuned model's responses on the test set, providing an average score to quantify performance.