Final Project: Fine-Tuning an R-Programming Model

Matthew White

April 28, 2025

Introduction

My initial idea was based on a project idea presented in class about fine-tuning a model to write code. Since my favorite programming language is R, I decided to fine-tune a model to write R code.

All of my code to download and run the model can be found here:

https://github.com/matthewwhite1/rprogrammingllama

Chosen Model and Data

Since I used Llama for a few of the assignments in this course, I decided to use some version of Llama. My laptop has an NVIDIA GeForce RTX 4060 GPU with 8 GB of RAM. So, one of the motivations behind my model choice was to choose one that would be able to fit on my GPU. The first iteration of my model used Llama 3.2-1B. When it became clear that this small version of Llama was not enough to make a coherent programmer, I moved up to Llama 3.2-3B. However, I still wasn't satisfied with the output of this model after fine-tuning.

After doing some more research on models, I found a version of Llama specifcially trained on programming called Code Llama [2]. As stated on its Huggingface page, Code Llama is a "code-specialized version of Llama 2 that was created by further training Llama 2 on its code-specific datasets." I decided that this would be a suitable choice for training my model to write code. Going along with my motivation on saving space, I chose the smallest version of Code Llama, which is the 7B model.

While Code Llama is already trained on code, I wanted to fine-tune it specifically to produce R code. So, I found a GitHub repository called GenCodeSearchNet that contains exactly what I wanted [1]. The repository contains data and code to produce results in a paper the authors wrote. The main draw for this repository for me is the "statcodesearch.json" file. This file contains a little over 1,000 examples of R code accompanied by code comments/explanations. This is the dataset I used to fine-tune my Code Llama model.

Training

My training architecture is very similar to what I used in a previous homework, just based on different data. I used QLoRA and Supervised Fine-Tuning for fine-tuning the model.

1. Load in the json file
2. Provide instructions for the model, which include instructions to ONLY respond with R code
3. Load the tokenizer
4. Preprocess the data by padding sequences to the same length and tokenizing
5. Set up the quantization
    - 4-bit quantization

    - bfloat16
6. Load in the Code Llama model
7. Set up the LoRA configuration
8. Define the training arguments
    - 2 epochs (to save on time)
    - Adam optimizer
9. Fine-tune the model using Supervised Fine-Tuning
10. Save the model

Front-End Interface

I used Gradio to create a front-end interface for interacting with my model. For the interface, I had to be very strict about what the model would output: "You are an expert R programmer. Only output valid R code. Do not include explanations, examples, or any extra text. Output R code only." Without this strict prompting, the model would often include more than just code, which isn't very helpful to a user that just wants R code.

The sampling of the model was performed with some of the following parameter values:

- max_new_tokens = 300, to stop the model from writing too much
- top_p = 0.9, to have the model pick the most likely options
- temperature = 0.7, to have the model not be too unpredictable
- repetition_penalty = 1.2, to penalize the model from repeating the same things

Evaluation

Rather than making a new evaluator file, I used the Gradio interface to evaluate the model's output. Throughout this testing, I had to make multiple changes to both the training code and the interface code. One problem I had was that after the first prompt, the model would respond fine with only code, and then by the time there was a second prompt, the model would freak out and write too much unrelated text. I had to change the model's internal prompt many times to get it to stop doing this, and even then, it still does it fairly often.

Another problem I encountered was the response code not being a good solution to the problem. Sometimes, I would ask the model to write something simple in R and it would create a good solution, and other times I would ask the model the same thing and the solution was not valid R code that could run without error.

I also had ChatGPT write some tests for the training and interface code, but they don't fully run.

Potential Future Work

If I had more time, I would like to retrain the model with a larger version of Code Llama. Fine-tuning on a version with more parameters may help the model provide better answers. Another thing I would like to do is find/generate more R data; 1,000 R examples may not be enough to have a fully coherent R programmer.

References

[1] Andor Diera, Abdelhalim Dahou, Lukas Galke, Fabian Karl, Florian Sihler, and Ansgar Scherp. Gencode-searchnet: A benchmark test suite for evaluating generalization in programming language understanding, 2023.

[2] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024.