Lecture 02: Tuning Techniques for Cost Optimization

Hamza Salem - Innopolis university

Agenda

Fine-Tuning and Customizability

Parameter-Efficient Fine-Tuning Methods

Cost and Performance Trade-Offs

Summary and Q&A

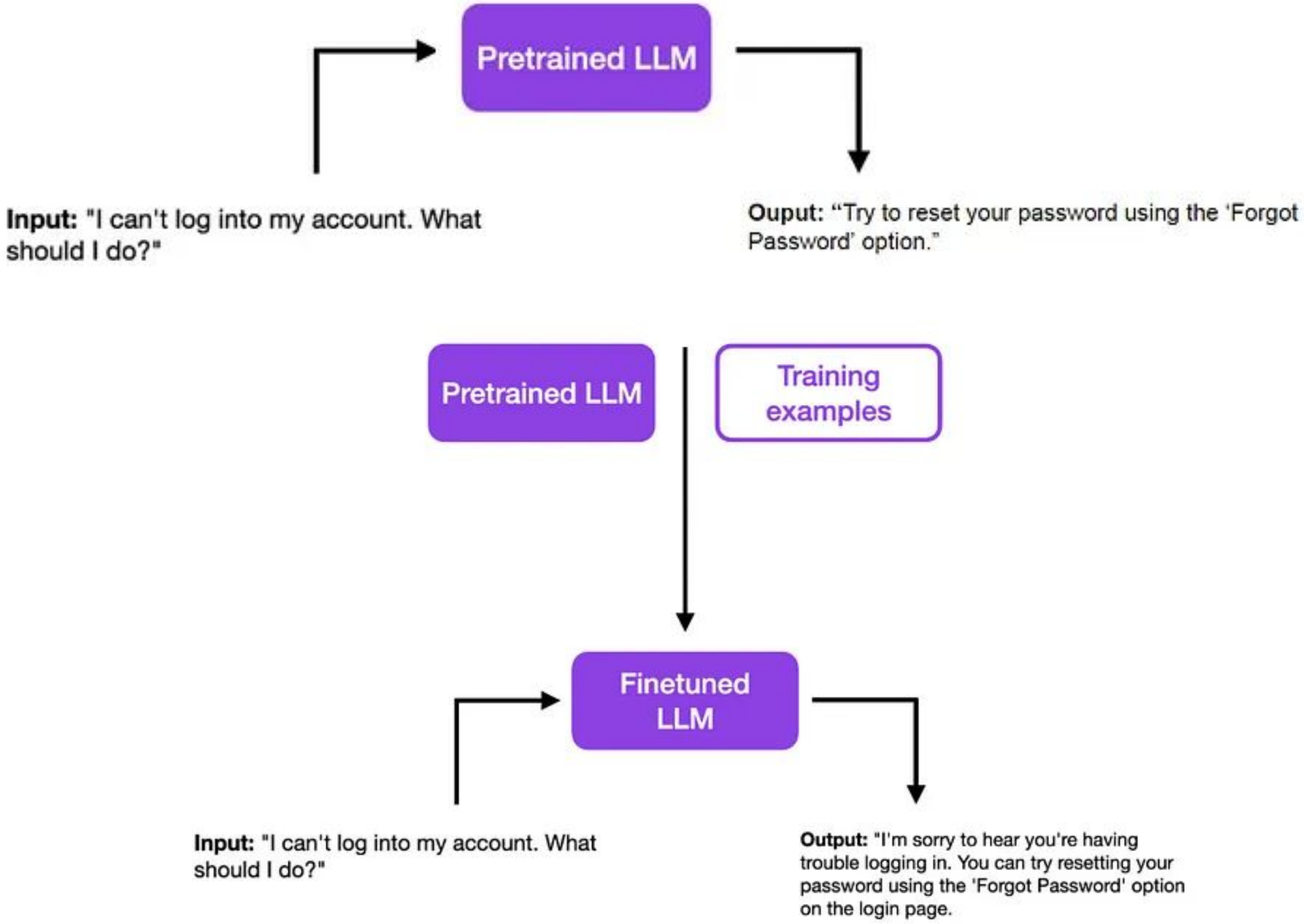
Fine-Tuning and Customizability

Fine-tuning involves adjusting a pre-trained LLM on a specific dataset to improve performance on a particular task.

Customizability: Allows you to tailor the model to specific needs (e.g., industry-specific terminology, domain-specific tasks).

Balance: Achieving a balance between general performance and customization is key.

Fine-tuning is like customizing the suit for specific missions, like underwater operations or stealth, enhancing its efficiency for that task.



Pretrained LLM

Input: "I can't log into my account. What should I do?"

Ouput: "Try to reset your password using the 'Forgot Password' option."

Pretrained LLM

Training examples

Finetuned LLM

Input: "I can't log into my account. What should I do?"

Output: "I'm sorry to hear you're having trouble logging in. You can try resetting your password using the 'Forgot Password' option on the login page.

Parameter-Efficient Fine-Tuning Methods

Why Parameter Efficiency?

Reducing resource consumption while maintaining performance.

Techniques:

- Distillation: Compressing a large model (teacher) into a smaller model (student) while retaining most of its performance.
- Pruning: Removing less significant parts of the model to reduce complexity.
- Quantization: Reducing the precision of the model's weights to save memory and processing power.

Distillation: A Closer Look

Teacher Model: A large, complex model that provides guidance.

Student Model: A smaller, distilled model that mimics the teacher’s outputs.

Steps Involved:

- Train the Teacher: Start with a high-performing model.
- Distill Knowledge: Train a smaller model on the predictions made by the teacher.
- Deploy the Student: Use the smaller model in production to save costs.

Distillation: A Closer Look

Teacher Model: A large, complex model that provides guidance.

Student Model: A smaller, distilled model that mimics the teacher’s outputs.

Steps Involved:

- Train the Teacher: Start with a high-performing model.
- Distill Knowledge: Train a smaller model on the predictions made by the teacher.
- Deploy the Student: Use the smaller model in production to save costs.

Real-world examples like BERT distilled to TinyBERT.

Pruning and Quantization

- Pruning: Trims the model by removing redundant parameters.

Example: Removing nodes in a neural network that contribute minimally to output.

- Quantization: Lowers precision of weights from 32-bit floats to 8-bit integers, reducing memory usage.

Example: Sacrificing some accuracy for faster processing and less storage.

Deploy in phone?

https://drive.google.com/file/d/1GGMqUtoSOroW87A4c98ZOy2r0JwhghZO/view?usp=sharing

Cost and Performance Implications

Performance vs. Cost: Understanding that reducing model size may affect accuracy or performance.

Scalability: Smaller models are easier to scale but might not perform as well on complex tasks.

Use-Cases: Deciding when to use parameter-efficient models (e.g., real-time applications vs. complex decision-making).

Key Takeaways

Fine-Tuning allows for customizability but requires careful balance.

Parameter-Efficient Methods like distillation and pruning help optimize costs without major losses in performance.

Trade-Offs: Every optimization technique has its pros and cons; understanding them is key to successful deployment.

Demo

https://cookbook.openai.com/examples/chat_finetuning_data_prep

https://colab.research.google.com/drive/1UnV2UZB4dY4FD6PsYJ6o8DtNS1zaY8Gv?usp=sharing

Exercise 1 : Data preparation

Use Google gemini to clean this data and translate it to english to generate it to be in this form:

```
{"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."}, {"role": "user", "content": "What's the capital of France?"}, {"role": "assistant", "content": "Paris, as if everyone doesn't know that already."}]}
```

- Submit the python code /ex1/main.py and file as data.txt or data.json

Exercise 2 : Fine-Tuning GPT

By using the data from previous exercise 1, extend this code to fine-tune GPT2.

- Your code should show how GPT2 will answer to a similar question from the dataset.

- Submit Python code /ex2/main.py.

Exercise 3: YTubeGPT - Optional -end of the day

Choose A youtube channel and get all transcripts for the videos and prepare this data for Tuning (GPT2 or GPT4), make it as telegram bot and deploy it and send it to @enghamzasalem in telegram.

+3 on the Course

Resources

- https://medium.com/mantisnlp/supervised-fine-tuning-customizing-llms-a2c1edbf22c3
- https://www.datacamp.com/blog/gpt-4o-mini
- https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results
- https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset
- https://velog.io/@minbrok/Gemini-Fine-Tuning-x51r7vws
- https://colab.research.google.com/github/GoogleCloudPlatform/generative-ai/blob/main/gemini/tuning/supervised_finetuning_using_gemini.ipynb#scrollTo=AVL2gfP-J5SL
- https://g4f.mintlify.app/docs/get-started/quickstart/install/pip