

Báo Cáo Kỹ Thuật:

Tinh Chỉnh LLM Qwen và LLama Trên
Tập Dữ Liệu Trắc Nghiệm Lập Trình
CodeMMLU

Người báo cáo: Trịnh Hoàng Hiệp

1. Giới thiệu




Báo cáo này tập trung vào việc tinh chỉnh Qwen và LLama hiệu quả, sau đó khám phá hiện tượng suy giảm độ chính xác khi tinh chỉnh các mô hình ngôn ngữ lớn (LLM) trên tập dữ liệu CodeMMLU, một benchmark được thiết kế để đánh giá khả năng hiểu ngôn ngữ lập trình và trả lời các tác vụ trắc nghiệm lập trình. Tôi kiểm tra các nguyên nhân tiềm ẩn của sự suy giảm độ chính xác khi fine-tune quá mức dẫn tới hiện tượng quên thảm khốc catastrophic forgetting (CF) và đưa ra các chiến lược để giảm thiểu tác động tiêu cực của nó.

2. Thiết Lập Thử Nghiệm

* Mô Hình:

Tôi đã tinh chỉnh một số LLM, bao gồm các mô hình mã nguồn mở và gọi các API của các mô hình nguồn đóng như GPT. Cụ thể:

Tôi tinh chỉnh các mô hình nguồn mở như Qwen và LLama vì hiệu suất của chúng được đánh giá rất tốt trên tiêu chí “CodeMMLU” được đề xuất bởi bài báo “A Multi-Task Benchmark for Assessing Code Understanding Capabilities of CodeLLMs”

#	Model	Syntactic Accuracy	Semantic Accuracy	Real-task Accuracy	CodeMMLU
1	 gpt-4o-2024-05-13	60.41	57.81	77.18	67
2	 Meta-Llama-3-70B-Instruct	64.9	62.96	60.84	62.45
3	 Meta-Llama-3.1-70B-Instruct	64.41	62.25	56.11	60
4	claude-3-sonnet@20240229	67.22	66.08	38.26	53.97
5	Qwen2-7B	58.31	55.23	49.3	53.28
6	Meta-Llama-3-8B	54.14	47.8	53.84	51.89

Ngoài ra tôi cũng gọi các API của gpt 4, gemini,... vì chúng được huấn luyện trên nguồn tri thức dồi dào và có hiệu suất rất tốt trên leaderboards.

Cuối cùng tôi thực hiện majority vote cho từng task_id dựa trên các file csv mà các mô hình sinh ra để đưa ra dự đoán cuối cùng.

* Tập Dữ Liệu:

Tôi đã sử dụng tập dữ liệu CodeMMLU, bao gồm nhiều câu hỏi trắc nghiệm lập trình trên các ngôn ngữ lập trình khác nhau.

* Phương Pháp Tinh Chỉnh:

Tôi đã thử nghiệm các kỹ thuật tinh chỉnh khác nhau, bao gồm tinh chỉnh hiệu quả tham số (PEFT), tinh chỉnh với learning rate động tuân theo chuỗi Chebyshev và gradient clipping.

LoRA

Cụ thể tôi đã sử dụng phương pháp LoRA: Low-Rank Adaptation để tinh chỉnh lại các LLM và đặt `max_grad_norm = 1.0`, để cắt những gradient quá lớn vì chúng có thể gây ra hiện tượng overfitting cũng như catastrophic forgetting khiến mô hình vừa mất đi tính tổng quát vừa có hiệu suất kém trên dữ liệu test.

Điều chỉnh learning rate theo chuỗi Chebyshev

Tôi sử dụng 1 Dynamic Learning Rate Scheduler giúp điều chỉnh learning rate theo chuỗi Chebyshev. Điều này giúp cho mô hình lúc đầu học được nhanh kiến trúc instruction và form dạng của output trong prompt khi train. Sau đó `weight_decay` sẽ tăng dần để model không quá khớp với dữ liệu mới mà quên đi các tri thức cũ, cũng như quá khớp vào dữ liệu train dẫn tới mất đi tính tổng quát trên tập test.

Tôi sử dụng một lịch trình learning rate thay đổi theo chuỗi Chebyshev có thể giúp điều chỉnh quá trình cập nhật trọng số theo một cách phi tuyến. Ý tưởng là dùng một lịch trình có dạng:

$$\text{lr}(t) = \frac{c_0}{2} + \sum_{n=1}^N \left[c_n \cos\left(\frac{2\pi n t}{T}\right) + d_n \sin\left(\frac{2\pi n t}{T}\right) \right]$$

với t là step hiện tại và T là tổng số step. Lịch trình như vậy có thể giúp làm chậm quá trình cập nhật khi đến những giai đoạn sau, từ đó bảo tồn kiến thức cũ.

* Chỉ Số Đánh Giá:

Tôi đã sử dụng độ chính xác làm chỉ số đánh giá chính, đo lường tỷ lệ các câu trả lời đúng mà mô hình đưa ra. Ngoài ra chúng tôi có kiểm thử hiệu suất thông qua các lần nộp bài trên kaggle

3. Kết Quả

Tôi quan sát thấy rằng trong một số trường hợp, độ chính xác của các LLM được tinh chỉnh trên tập dữ liệu CodeMMLU thấp hơn so với các mô hình ít được tinh chỉnh. Điều này đặc biệt đúng đối với các mô hình tinh chỉnh trên toàn bộ tập dữ liệu sẽ có hiệu suất thấp hơn các mô hình chỉ tinh chỉnh trên 20-30% tập dữ liệu huấn luyện.

Model	Lượng dữ liệu train	Độ chính xác
Qwen 2.5 B	0.2 epoch	0.64
Qwen 2.5 B	1 epoch	0.55
LLama 3 8B	0.66 epoch	0.54
LLama 3 8B	1 epoch	0.48

Ngoài ra tại tác vụ khác mô hình đã bị quên
Cụ thể tại task dự đoán số fibonacci tiếp theo mô hình chưa train có thể dự đoán đúng số tiếp theo là 13:

```
_ = model.generate(**inputs, streamer = text_streamer, max_new_tokens = 128)

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that completes the instruction.

### Instruction:
Continue the fibonacci sequence.

### Input:
1, 1, 2, 3, 5, 8

### Response:
13<|endoftext|>
```

Trong khi mô hình sau khi fine tune toàn bộ 1 epoch dự đoán sai

```
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that completes the instruction.

### Instruction:
Continue the fibonacci sequence.

### Input:
1, 1, 2, 3, 5, 8

### Response:
2<|endoftext|>
```

Các Nguyên Nhân Tiềm Ẩn Của Sự Suy Giảm Độ Chính Xác

- * Quá khớp: Tập dữ liệu CodeMMLU có thể không đủ lớn hoặc đủ đa dạng để ngăn ngừa quá khớp, đặc biệt là khi tinh chỉnh các mô hình lớn hơn.
- * Quên thảm khốc: Tinh chỉnh các LLM trên tập dữ liệu CodeMMLU có thể dẫn đến việc quên thảm họa các kiến thức lập trình đã học trước đó.
- * Độ phức tạp của tác vụ: Các câu hỏi trắc nghiệm lập trình trong CodeMMLU có thể yêu cầu khả năng suy luận và hiểu biết sâu sắc về các khái niệm lập trình, điều này có thể khó khăn đối với các LLM ngay cả sau khi tinh chỉnh.

Chiến Lược Để Giảm Thiểu Sự Suy Giảm Độ Chính Xác

- * Tinh chỉnh hiệu quả tham số (PEFT): Sử dụng các kỹ thuật PEFT, chẳng hạn như LoRA hoặc QLoRA, để giảm nguy cơ quên thảm họa và quá khớp.
- * Thiết kế prompt: Thiết kế các prompt được xây dựng tốt để hướng dẫn mô hình đến câu trả lời đúng.

* Sử dụng kỹ thuật của học liên tục như KD (knowledge distillation) để chất lọc thông tin từ mô hình gốc nếu như dự đoán của mô hình teacher đã chính xác rồi.

* Dynamic Learning Rate Scheduler giúp điều chỉnh learning rate theo chuỗi Chebyshev

* Kết hợp các phương pháp: Sử dụng kết hợp các phương pháp trên cùng với gọi các API của mô hình đóng để tận dụng các tri thức không bị quên thảm khốc để đạt được kết quả tốt nhất.

Kết quả:

Đối với Qwen2.5 khi tôi sử dụng các kỹ thuật như Lora+ Chebyshev độ chính xác tăng từ 0.52 lên 0.64 khi fine tune trên 0.2 epoch

Sau đó sử dụng kỹ thuật KD tinh chỉnh trên 1 epoch toàn bộ tập trainset độ chính xác chỉ giảm nhẹ còn 0.62 chứ không giảm mạnh xuống 0.55 như báo cáo trong bảng nữa. Điều này cho thấy những kỹ thuật đề xuất hiệu quả khi fine tune LLM

4. Kết luận

Việc suy giảm độ chính xác khi tinh chỉnh LLM trên tập dữ liệu CodeMMLU là một vấn đề phức tạp đòi hỏi phải xem xét cẩn thận. Bằng cách hiểu các nguyên nhân tiềm ẩn và áp dụng các chiến lược thích hợp, chúng ta có thể giảm thiểu tác động tiêu cực của nó và cải thiện hiệu suất của các LLM được tinh chỉnh cho các tác vụ trắc nghiệm lập trình.

5. Hướng Nghiên Cứu Tương Lai

* Khám phá các phương pháp tinh chỉnh mới được thiết kế đặc biệt cho các tác vụ lập trình.

* Phát triển các chỉ số đánh giá mới nắm bắt tốt hơn hiệu suất của LLM trong các tác vụ lập trình.

* Nghiên cứu tác động của việc tăng cường dữ liệu và các kỹ thuật PEFT đối với hiệu suất của LLM trên tập dữ liệu CodeMMLU.

* Điều tra khả năng sử dụng LLM để tạo các câu hỏi trắc nghiệm lập trình mới cho tập dữ liệu CodeMMLU.

* Tăng cường dữ liệu: Tăng kích thước và tính đa dạng của tập dữ liệu CodeMMLU bằng cách sử dụng các kỹ thuật tăng cường dữ liệu.

* Kỹ thuật học đa nhiệm: Tinh chỉnh các LLM trên nhiều tập dữ liệu lập trình khác nhau để cải thiện khả năng khái quát hóa.

* Tinh chỉnh theo giai đoạn: Chia quá trình tinh chỉnh thành nhiều giai đoạn, tập trung vào các khía cạnh khác nhau của khả năng lập trình