

Xây dựng Chatbot giải toán

Dinh-Thang Duong, Minh-Duc Bui và Quang-Vinh Dinh

Proof reading team: Minh-Châu Phạm, Hoàng-Nguyên Vũ và Đăng-Nhã Nguyễn

Ngày 21 tháng 1 năm 2024

Phần I: Giới thiệu

Chatbot (Large Language Models): Chatbot là một trong những ứng dụng phổ biến nhất trong lĩnh vực Trí tuệ nhân tạo (AI). Điều này càng được thấy rõ hơn với sự ra đời của các Mô hình ngôn ngữ lớn (LLMs) trong thời gian qua với khả năng thực hiện rất nhiều các tác vụ khác nhau với độ chính xác cao và vẫn đang được phát triển, cải tiến. Sự tiến bộ trong công nghệ này không chỉ cải thiện chất lượng tương tác giữa người và máy, mà còn mở ra những khả năng mới trong việc tự động hóa và cá nhân hóa các dịch vụ.



You

xin chào



ChatGPT

Xin chào! Bạn cần tôi giúp gì hôm nay?



You

tôi cần bạn hỗ trợ tôi học lập trình python



ChatGPT

Tất nhiên, tôi rất vui lòng hỗ trợ bạn học lập trình Python. Python là một ngôn ngữ lập trình phổ biến và mạnh mẽ, tốt cho cả người mới bắt đầu và lập trình viên có kinh nghiệm. Bạn có muốn bắt đầu từ cơ bản, hay bạn đã có kinh nghiệm lập trình và muốn học những thứ cụ thể hơn?



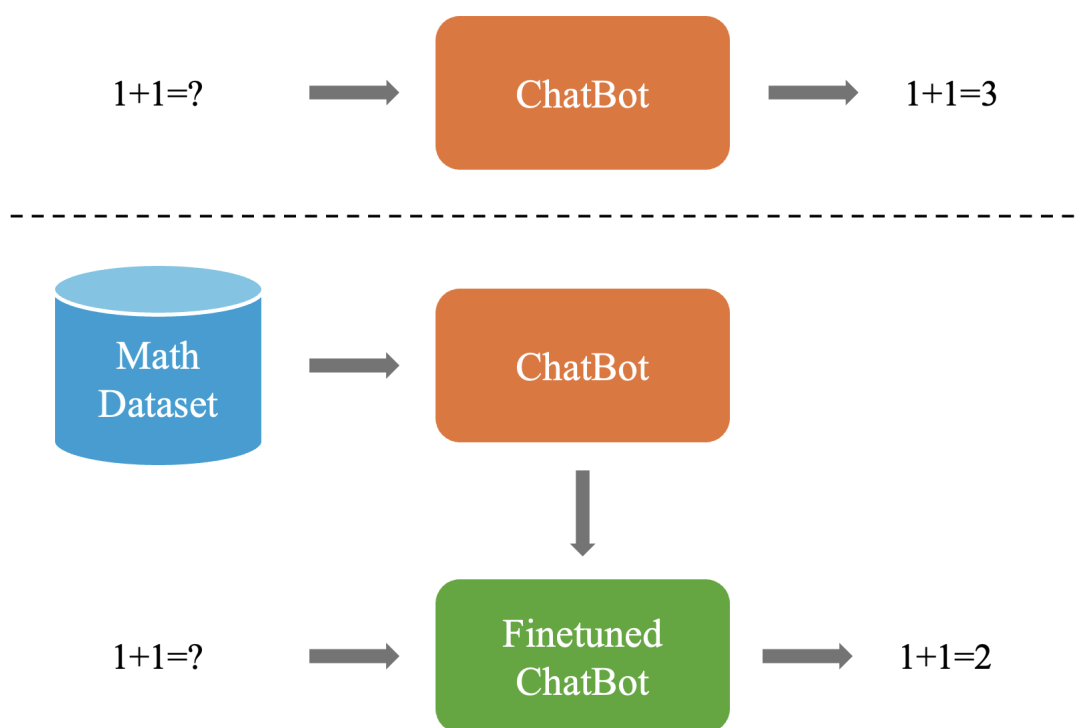
Hình 1: Ảnh minh họa về ChatGPT, một ứng dụng liên quan đến Chatbot sử dụng Mô hình ngôn ngữ lớn (LLMs).

Cụ thể, với Chatbot, chúng ta cần xây dựng một chương trình có thể tương tác được với người dùng

bằng văn bản, thường sẽ được diễn ra trong một hộp tin nhắn thoại. Như vậy, chương trình mà chúng ta xây dựng trong project này sẽ có Input và Output như sau:

- **Input:** Một đoạn văn bản với nội dung bất kì (prompt).
- **Output:** Một đoạn văn bản phản hồi từ Chatbot (response).

Bên cạnh xây dựng một chương trình Chatbot có thể trả lời các hội thoại thường nhật, chúng ta cũng sẽ mở rộng tìm hiểu thêm về cách làm cho Chatbot sử dụng mô hình ngôn ngữ lớn có thể hoạt động tốt trên một công việc cố định nào đó. Cụ thể, chúng ta làm cho Chatbot cải thiện khả năng giải toán tiểu học thông qua việc fine-tuning trên bộ dữ liệu [vi_grade_school_math_mcq](#), đây là bộ data về các bài tập toán từ lớp 1 đến 5 bằng tiếng Việt.



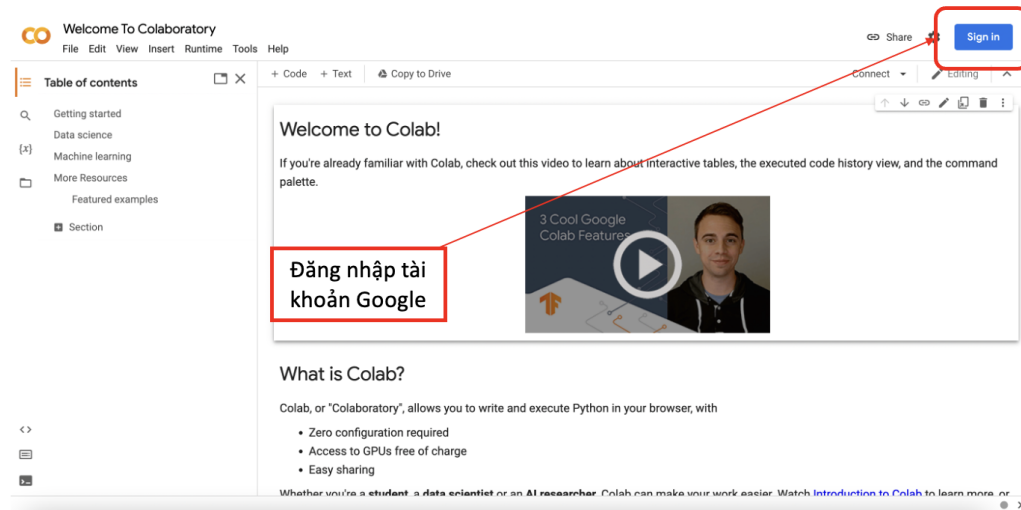
Hình 2: Minh họa về việc fine-tuning Mô hình ngôn ngữ lớn trên bộ dữ liệu về toán

Phần II: Nội dung

A. How to use Google Colab

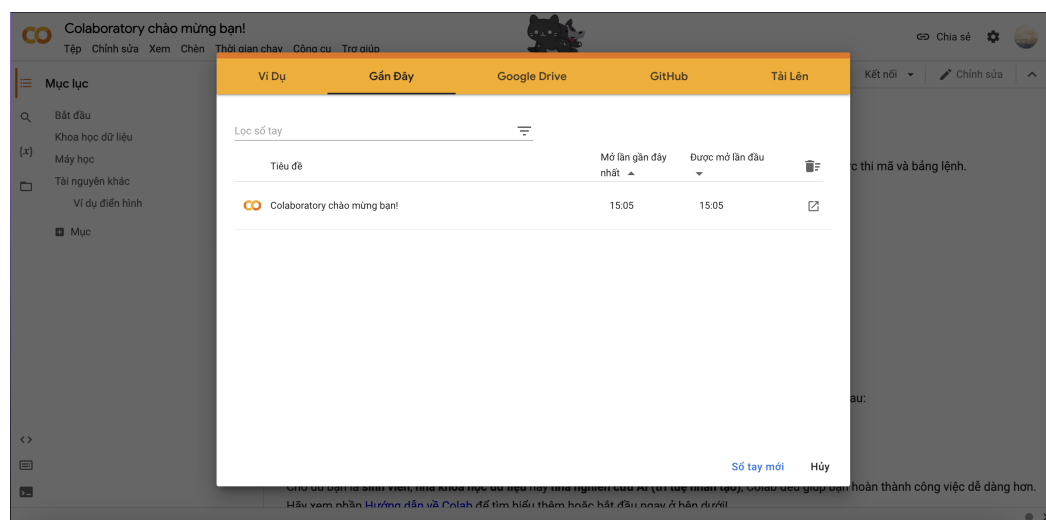
Để thuận tiện trong việc huấn luyện các mô hình Trí tuệ nhân tạo (AI), chúng ta sẽ dùng Google Colab làm môi trường cài đặt. Các bước sử dụng Google Colab được thực hiện như sau:

- **Bước 1:** Truy cập vào đường dẫn sau: [link](#). Nếu truy cập thành công, các bạn sẽ thấy giao diện như hình dưới đây:



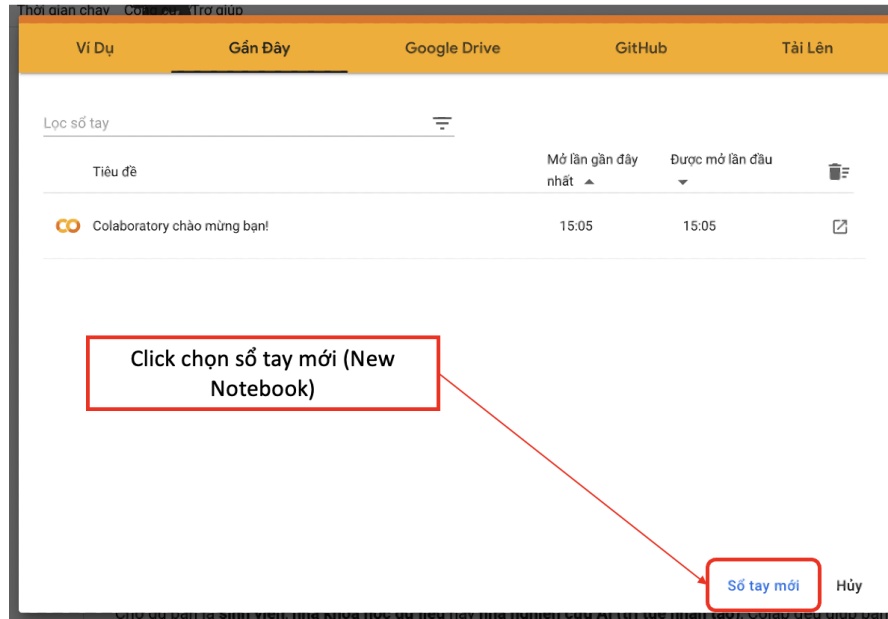
Hình 3: Giao diện chính của Google Colab

Sau đó, các bạn hãy đăng nhập bằng tài khoản Google của mình. Nếu đăng nhập thành công, một cửa sổ mới sẽ hiện lên như hình dưới đây:



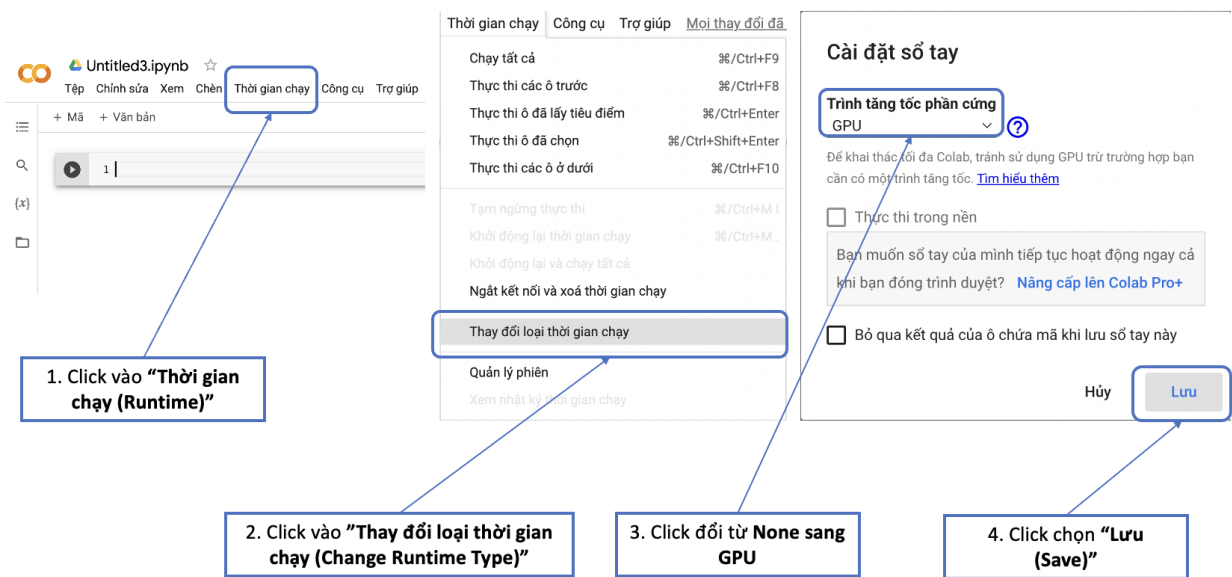
Hình 4: Giao diện Google Colab sau khi đăng nhập thành công

- **Bước 2:** Khởi tạo notebook mới. Notebook sẽ là giao diện để ta có thể viết các dòng lệnh Python. Để tạo được notebook, các bạn thực hiện thao tác theo hình dưới đây:



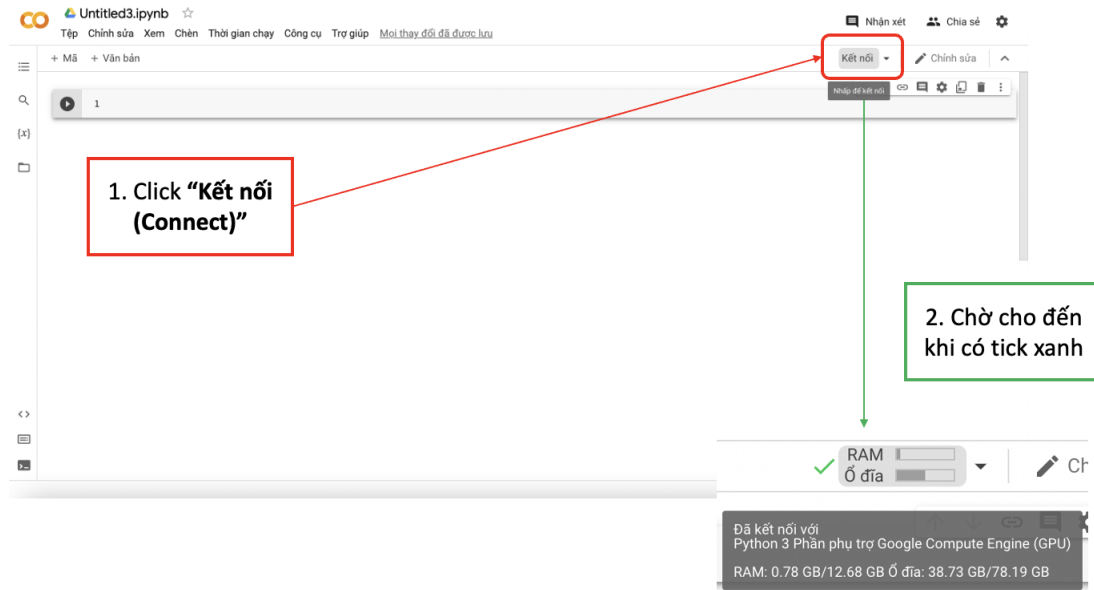
Hình 5: Khởi tạo notebook mới

- **Bước 3:** Thay đổi runtime của notebook từ CPU thành GPU.



Hình 6: Các bước kích hoạt GPU cho notebook mới trên Google Colab

- **Bước 4:** Cuối cùng, để có thể thực thi các dòng lệnh Python, ta cần khởi động notebook. Các thao tác khởi động một notebook trong Google Colab sẽ như hình sau:



Hình 7: Khởi động một notebook có GPU trong Google Colab

Sau khi thực hiện các bước trên, các bạn đã có một môi trường code Python với GPU miễn phí từ Google.

B. Fine-tuning VinaLLaMA for Chatbot

Trong phần này, chúng ta sẽ xây dựng một Chatbot sử dụng Mô hình ngôn ngữ lớn được huấn luyện chủ yếu trên bộ dữ liệu tiếng Việt là VinaLLaMA. Các bước thực hiện như sau:

1. **Cài đặt thư viện:** Chúng ta sẽ cần cài đặt một số thư viện sau để có thể chạy được một mô hình ngôn ngữ lớn từ thư viện HuggingFace:

```
1 !pip install -q -U bitsandbytes
2 !pip install -q -U datasets
3 !pip install -q -U git+https://github.com/huggingface/transformers.git
4 !pip install -q -U git+https://github.com/huggingface/peft.git
5 !pip install -q -U git+https://github.com/huggingface/accelerate.git
6 !pip install -q -U loralib
7 !pip install -q -U einops
8 !pip install -q -U googletrans==3.1.0a0
```

2. **Import các thư viện cần thiết:** Sau khi đã tải xong, chúng ta sẽ thực hiện import các thư viện đã tải cũng như một số thư viện khác để phục vụ cho chương trình:

```
1 import json
2 import os
3 import bitsandbytes as bnb
4 import torch
5 import torch.nn as nn
6 import transformers
7
8 from googletrans import Translator
9 from pprint import pprint
10 from datasets import load_dataset
11 from huggingface_hub import notebook_login
12 from peft import (
13     LoraConfig,
14     PeftConfig,
15     PeftModel,
16     get_peft_model,
17     prepare_model_for_kbit_training
18 )
19 from transformers import (
20     AutoConfig,
21     AutoModelForCausalLM,
22     AutoTokenizer,
23     BitsAndBytesConfig
24 )
25
26 os.environ["CUDA_VISIBLE_DEVICES"] = "0"
```

3. **Khởi tạo mô hình:** Mô hình ngôn ngữ lớn mà chúng ta sẽ sử dụng trong bài này có tên gọi là VinaLLaMA, một mô hình được nhóm tác giả người Việt thực hiện huấn luyện trên bộ dữ liệu chủ yếu về tiếng Việt. Để khởi tạo mô hình lên trên file notebook, chúng ta sẽ chạy đoạn code sau:

```
1 MODEL_NAME = "vilm/vinallama-7b-chat"
2
3 bnb_config = BitsAndBytesConfig(
4     load_in_4bit=True,
5     bnb_4bit_use_double_quant=True,
6     bnb_4bit_quant_type="nf4",
7     bnb_4bit_compute_dtype=torch.bfloat16
```

```

8 )
9
10 model = AutoModelForCausalLM.from_pretrained(
11     MODEL_NAME,
12     device_map="auto",
13     trust_remote_code=True,
14     quantization_config=bnb_config
15 )
16
17 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
18 tokenizer.pad_token = tokenizer.eos_token
19
20 model.gradient_checkpointing_enable()
21 model = prepare_model_for_kbit_training(model)
22
23 config = LoraConfig(
24     r=16,
25     lora_alpha=32,
26     target_modules=[
27         "q_proj",
28         "up_proj",
29         "o_proj",
30         "k_proj",
31         "down_proj",
32         "gate_proj",
33         "v_proj"
34     ],
35     lora_dropout=0.05,
36     bias="none",
37     task_type="CAUSAL_LM"
38 )
39
40 model = get_peft_model(model, config)

```

Trong đó:

- **Dòng 1:** Khai báo biến chứa tên của mô hình ngôn ngữ lớn chúng ta mong muốn sử dụng, ở đây sẽ là ViniLLaMA phiên bản 7b-chat.
 - **Dòng 3 - 40:** Khởi tạo mô hình và các cài đặt cần thiết.
4. **Sử dụng mô hình (trước khi huấn luyện):** Mô hình vừa khởi tạo đã được nhóm tác giả huấn luyện trên một bộ dữ liệu rất lớn, có thể thực hiện được nhiều task khác nhau. Chúng ta có thể tương tác với mô hình ngay lúc này, bằng cách viết một đoạn chat mô tả mệnh lệnh nào đó (prompt) và gửi vào mô hình. Sau một khoảng thời gian tính toán, mô hình sẽ trả về câu trả lời phù hợp. Cách thực hiện như sau:

- (a) Cài đặt một vài tham số cần thiết cho mô hình, các tham số này sẽ ảnh hưởng đến kết quả trả lời của mô hình ngôn ngữ lớn:

```

1 generation_config = model.generation_config
2 generation_config.max_new_tokens = 200
3 generation_config.temperature = 0.7
4 generation_config.top_p = 0.7
5 generation_config.num_return_sequences = 1
6 generation_config.pad_token_id = tokenizer.eos_token_id
7 generation_config.eos_token_id = tokenizer.eos_token_id

```

- (b) **Khai báo prompt:** Chúng ta sẽ khởi tạo một biến chứa đoạn prompt, câu mệnh lệnh hoặc một đoạn tin nhắn mà chúng ta muốn gửi vào mô hình. Cụ thể trong VinaLLaMA, chúng ta sẽ có format cố định cho đoạn prompt như sau:

```

1 prompt = """
2 <|im_start|>system
3 Bạn là một trợ lí AI hữu ích. Hãy trả lời người dùng một cách chính xác.
4 <|im_end|>
5 <|im_start|>user
6 {your_task}
7 <|im_end|>
8 <|im_start|>assistant
9 """.strip()

```

Trong đó, `{your_task}` là một đoạn văn bản mô tả một nhiệm vụ, câu hỏi hay một câu nói bất kì mà bạn mong muốn gửi đến mô hình.

Dựa vào format trên, ta có thể thử đặt một yêu cầu xây dựng một hàm Python cho mô hình như trong môi trường code sau:

```

1 prompt = """
2 <|im_start|>system
3 Bạn là một trợ lí AI hữu ích. Hãy trả lời người dùng một cách chính xác.
4 <|im_end|>
5 <|im_start|>user
6 Viết một hàm tính tổng hai số trong python
7 <|im_end|>
8 <|im_start|>assistant
9 """.strip()

```

- (c) **Chạy mô hình:** Sử dụng đoạn code dưới đây, ta đưa đoạn prompt đã khởi tạo để lấy câu trả lời từ mô hình như sau:

```

1 %%time
2 device = 'cuda' if torch.cuda.is_available() else 'cpu'
3
4 encoding = tokenizer(prompt, return_tensors="pt").to(device)
5 with torch.inference_mode():
6     outputs = model.generate(
7         input_ids=encoding.input_ids,
8         attention_mask=encoding.attention_mask,
9         generation_config=generation_config
10    )
11
12 print(tokenizer.decode(outputs[0], skip_special_tokens=True))

```

Khi quá trình tính toán hoàn tất, ta nhận được kết quả in ra màn hình là câu trả lời của mô hình ứng với đoạn prompt:


```

<|im_start|> system
Bạn là một trợ lý AI hữu ích. Hãy trả lời người dùng một cách chính xác.

<|im_start|> user
Viết một hàm tính tổng hai số trong python

<|im_start|> assistant
Dưới đây là một hàm Python nhận hai số làm đầu vào và trả về tổng của chúng:

```python
def add_two_numbers(a, b):
 return a + b

Ví dụ sử dụng
result = add_two_numbers(5, 3)
print(result) # Kết quả: 8
```

Hàm này, `add_two_numbers`, nhận hai tham số, `a` và `b`, đại diện cho hai số bạn muốn cộng. Nó sử dụng toán tử `+`
CPU times: user 20.3 s, sys: 231 ms, total: 20.5 s
Wall time: 20.6 s


```

Hình 8: Ví dụ về câu trả lời của mô hình về việc viết một hàm Python ứng với mô tả trong prompt

Như vậy, có thể thấy chỉ với mô hình gốc (gọi là pre-trained model), chúng ta đã có thể tương tác với mô hình ngôn ngữ lớn và yêu cầu thực hiện một tác vụ nào đó với độ chính xác tương đối. Trong lĩnh vực Machine Learning, chúng ta còn có thể cải thiện kết quả của pre-trained model với một task cụ thể nào đó bằng cách áp dụng một kỹ thuật được gọi là fine-tuning. Cụ thể, chúng ta sẽ tiếp tục thực hiện huấn luyện mô hình, trên một bộ dữ liệu với các task cụ thể hơn (ứng với nhu cầu và mục đích sử dụng của chúng ta).

- Tải bộ dữ liệu fine-tuning:** Trong bài này, vì mục tiêu của chúng ta là xây dựng một Chatbot có thể trò chuyện những cuộc đối thoại thường nhật, nên ở phần sau chúng ta sẽ thực hiện fine-tuning ViniLLaMA trên bộ dữ liệu hội thoại để cải thiện chất lượng câu trả lời. Ta thực hiện tải bộ dữ liệu có tên là `alespalla/chatbot_instruction_prompts` như sau:

```
1 data = load_dataset('alespalla/chatbot_instruction_prompts')
```

| response
string · lengths | prompt
string · lengths |
|---|---|
|  |  |
| . Determine the cause of the clog. Clogs in kitchen drains can be caused by food, grease, soap scum, and... | What's the best way to fix my kitchen drain? 1 |
| I have broken something by accident in the past as well, so I definitely empathize with you about the... | I'm just going to go to the store and whatever will happen it's going to happen. |
| Bonjour | Translate the phrase "Good Morning" to French |
| Samurai warriors played an incredibly important role in the development of Japanese culture. Samurai... | What contribution did Samurai warriors make to Japanese culture? |
| I think both you and your friend will feel better about the situation if you're just honest. Both of yo... | Yeah, maybe you're right. |
| Step 1: Gather all the ingredients you will need for the aioli: 1/2 cup of mayonnaise, 1 garlic clove... | I am cooking for guests tomorrow and need to know how to make aioli. Can you give me a step-by-step recipe. |

Hình 9: Minh họa một số mẫu dữ liệu trong bộ dữ liệu vừa tải về

Trong đoạn code, ta sử dụng hàm `load_data()` từ thư viện **datasets**, hàm này cho phép tải các bộ dữ liệu trong database của thư viện. Bộ dữ liệu được ta lưu vào biến `data`, khi in biến này, ta có thông tin như sau:

```

1 data
DatasetDict({
  train: Dataset({
    features: ['response', 'prompt'],
    num_rows: 258042
  })
  test: Dataset({
    features: ['response', 'prompt'],
    num_rows: 64511
  })
})

```

Hình 10: Dữ liệu của biến `data`. Biến có kiểu dữ liệu là **DatasetDict**, một kiểu dữ liệu riêng biệt của thư viện **datasets**

Ở đây, chúng ta có thể hình dung biến `data` lúc này là một dictionary chứa hai key là 'train' và 'test'. Đây là hai bộ dữ liệu, chứa list các thông tin về câu prompt của người dùng và câu trả lời của mô hình tương ứng.

6. **Xây dựng bộ dữ liệu fine-tuning:** Với bộ dữ liệu đã tải, chúng ta sẽ sử dụng để thực hiện fine-tuning mô hình, tức sẽ huấn luyện cho mô hình học thêm các dữ liệu từ bộ dữ liệu mới này. Các bước làm như sau:

- (a) **Xây dựng hàm tạo prompt:** Trong trường hợp huấn luyện VinaLLaMA, chúng ta cần thay đổi dữ liệu vào đúng format prompt như ở phần trước. Nhận thấy trong format prompt, ô user sẽ nhận input của người dùng, ứng với trường "prompt" của bộ dữ liệu. Ô assistant là câu trả lời của mô hình, ứng với trường "response" của bộ dữ liệu. Vì vậy, ta sẽ xây dựng hàm để đưa vào đúng khuôn format như sau:

```

1 def generate_prompt(data_point):
2     translator = Translator()
3     vn_prompt = translator.translate(data_point['prompt'], src='en', dest='vi')
4     vn_response = translator.translate(data_point['response'], src='en', dest='vi')
5
6     return f"""
7     <|im_start|>system
8     Bạn là một trợ lí AI hữu ích. Hãy trả lời người dùng một cách chính xác.
9     <|im_end|>
10    <|im_start|>user
11    {vn_prompt}
12    <|im_end|>
13    <|im_start|>assistant
14    {vn_response}
15    """.strip()

```

- (b) **Xây dựng hàm tokenization:** Đối với bất kì mô hình ngôn ngữ lớn nào, để xử lý một văn bản nào, trước hết chúng ta cần thực hiện tokenization lên văn bản đó. Hiểu một cách đơn giản, chúng ta sẽ đưa văn bản từ dạng string thành một list (vector) các con số:

Ở đây, ta sẽ thiết kế hàm tạo câu prompt với điểm dữ liệu gồm cặp (response, prompt) đầu vào, sau đó thực hiện tokenize câu prompt, code cài đặt như sau:

```

1 def generate_and_tokenize_prompt(data_point):
2     full_prompt = generate_prompt(data_point)
3     tokenized_full_prompt = tokenizer(full_prompt, padding=True, truncation=
4         True)
5     return tokenized_full_prompt

```

(c) **Áp dụng tokenization vào bộ dữ liệu:** Với hàm vừa xây dựng, ta áp vào biến data vừa rồi để thực hiện lời gọi hàm lên từng điểm dữ liệu trong biến data. Cách làm như sau:

```

1 data = data['train'].shard(num_shards=50, index=0).filter(lambda sample:
2     sample['response'] != '' and sample['prompt'] != '').shuffle().map(
3     generate_and_tokenize_prompt)

```

Cách làm này được thực hiện với sự hỗ trợ của kiểu dữ liệu **DatasetDict**.


7. Thực hiện huấn luyện mô hình (fine-tuning): Sau khi đã chuẩn bị xong bộ dữ liệu hoàn tất, chúng ta bắt đầu huấn luyện mô hình ngôn ngữ lớn, chạy các dòng lệnh sau:

```

1 training_args = transformers.TrainingArguments(
2     per_device_train_batch_size=1,
3     gradient_accumulation_steps=4,
4     num_train_epochs=1,
5     learning_rate=2e-4,
6     fp16=True,
7     save_total_limit=3,
8     logging_steps=1,
9     output_dir="experiments",
10    optim="paged_adamw_8bit",
11    lr_scheduler_type="cosine",
12    warmup_ratio=0.05,
13 )
14
15 trainer = transformers.Trainer(
16     model=model,
17     train_dataset=data,
18     args=training_args,
19     data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=
20     False)
21 )
22 model.config.use_cache = False
23 trainer.train()

```

Khi các bạn thấy bảng dưới đây xuất hiện, điều đó chứng tỏ tiến trình huấn luyện đã bắt đầu thành công, việc còn lại của chúng ta sẽ chỉ cần chờ cho tới khi việc thực thi hoàn tất.



| Step | Training Loss |
|------|---------------|
| 1 | 4.489600 |
| 2 | 3.980300 |
| 3 | 2.588000 |
| 4 | 2.978800 |
| 5 | 3.521400 |
| 6 | 3.824300 |
| 7 | 4.167600 |
| 8 | 3.548400 |
| 9 | 4.254500 |
| 10 | 3.988800 |

Hình 11: Ảnh minh họa bảng hiển thị các thông tin trong quá trình thực hiện huấn luyện mô hình ngôn ngữ lớn

8. **Chạy mô hình đã fine-tuning:** Cuối cùng, ta sẽ thử tương tác với mô hình sau khi đã được fine-tuning như sau:

```

1 %%time
2 device = 'cuda' if torch.cuda.is_available() else 'cpu'
3
4 prompt = ""
5 <|im_start|>system
6 Bạn là một trợ lí AI hữu ích. Hãy trả lời người dùng một cách chính xác.
7 <|im_end|>
8 <|im_start|>user
9 Mô tả về thành phố Đà Lạt
10 <|im_end|>
11 <|im_start|>assistant
12 """.strip()
13
14 encoding = tokenizer(prompt, return_tensors="pt").to(device)
15 with torch.inference_mode():
16     outputs = model.generate(
17         input_ids=encoding.input_ids,
18         attention_mask=encoding.attention_mask,
19         generation_config=generation_config
20     )
21
22 print(tokenizer.decode(outputs[0], skip_special_tokens=True))

```

Kết quả trả về của mô hình cho câu prompt trên được mô tả như hình dưới đây:

<|im_start|> system

Bạn là một trợ lý AI hữu ích. Hãy trả lời người dùng một cách chính xác.

<|im_start|> user

Mô tả về thành phố Đà Lạt

<|im_start|> assistant

Đà Lạt là một thành phố đẹp như tranh vẽ nằm ở vùng cao nguyên của tỉnh Lâm Đồng, Việt Nam. Được biết đến như "Thành phố ngàn hoa", Đà Lạt nổi tiếng với những khu vườn tươi tốt, những con đường đẹp như tranh vẽ và những ngôi nhà đầy màu sắc. Thành phố này có khí hậu mát mẻ, với nhiệt độ trung bình từ 15 đến 20 độ C vào những tháng mùa đông. Vào những tháng mùa hè, nhiệt độ có thể lên tới 25 độ C. Đà Lạt nổi tiếng với các món ăn ngon, từ các món ăn truyền thống Việt Nam đến các món ăn quốc tế. Thành phố cũng nổi tiếng với các lễ hội và sự kiện văn hóa, chẳng hạn như Lễ hội hoa hồng, Lễ hội mùa đông và Lễ hội mùa xuân. Đà Lạt là một điểm đến tuyệt vời cho những người yêu thích thiên nhiên, những người đam mê ẩm thực và những người tìm kiếm một nơi yên bình để thư giãn.

Hình 12: Câu trả lời của mô hình, phần văn bản được bôi đỏ là phản hồi của mô hình về yêu cầu "Mô tả về thành phố Đà Lạt" của người dùng

C. Fine-tuning VinaLLaMA for solving elementary math

Chatbot hiện tại của chúng ta, mặc dù có một khả năng hội thoại tương đối ổn. Song khi thực hiện giải quyết các bài toán tiểu học tưởng chừng như rất đơn giản, lại gây khó khăn cho Chatbot. Đây chính là vấn đề chung của các Chatbot sử dụng mô hình ngôn ngữ bởi hạn chế trong việc xử lý các tác vụ đòi hỏi sự suy luận logic và tính toán. Với pre-trained VinaLLaMA, ta có thể thấy rõ điều này ở ví dụ dưới đây:

```
<|im_start|> system Bạn là một chuyên gia về toán. Bạn sẽ nhận câu hỏi trắc nghiệm kèm theo các lựa chọn, hãy giải step by step nếu có và chọn phương án đúng.
```

```
<|im_start|> user
### Câu hỏi: Số gồm 1 đơn vị và 2 chục đọc là :
### Các lựa chọn:
A. 20
B. 21
C. 30
D. 31
### Câu trả lời:
```

```
<|im_start|> assistant
Để giải quyết vấn đề này, chúng ta cần hiểu cách đọc số gồm 1 đơn vị và 2 chục. Một chục bằng 10, vì vậy nếu chúng ta có 2 chục, chúng ta có  $2 * 10 = 20$  đơn vị. Bây giờ, chúng ta cần cộng thêm 1 đơn vị. Do đó, số đọc là  $20 + 1 = 21$ . Bây giờ, hãy kiểm tra các lựa chọn đã cho: A. 20 B. 21 C. 30 D. 31 Phương án đúng là B. 21. Vậy nên, số gồm 1 đơn vị và 2 chục đọc là 21.
```

Hình 13: Kết quả giải một bài toán tiểu học của pre-trained VinaLLaMA. Có thể thấy sự suy luận cũng như tính toán của mô hình đã bị sai

Để giải quyết vấn đề này, chúng ta sẽ tiếp tục áp dụng kỹ thuật fine-tuning trên một bộ dữ liệu về trắc nghiệm toán cấp 1 của Việt Nam. Đây là bộ dữ liệu được thu tập tại [hlj/vi_grade_school_math_mcq](https://huggingface.co/datasets/hlj/vi_grade_school_math_mcq). Tổng quan về bộ dữ liệu như sau:

| grade | id | title | problems | url |
|------------------|-----------------------------------|---|---|--|
| string · classes | string · lengths | string · lengths | list | string · lengths |
| 5 values | 32 | 16 | | 62 |
| | 32 | 117 | | 160 |
| 1 | 34a7a20a1cec28e5a0275dec1c9a245e | Bài tập ôn hè Toán lớp 1 Chuyển đề 5: Đo độ dài ... | [{ "choices": ["A. 10 cm", "B. 8 cm", "C. 9 cm", "D. 7 cm"], "explanation": "Hướng dẫn giải \n Đá..." | https://khoaoc.vietjack.com/thi-online/bai-tap-on-he-toan-lop-1-... |
| 1 | 490508bdbc190af20a2cd30153ee1f0a | Bài tập ôn hè Toán lớp 1 Chuyển đề 3: Bài toán c... | [{ "choices": ["A. 5 quả", "B. 4 quả", "C. 1 quả", "D. 9 quả"], "explanation": "Hướng dẫn giải..." | https://khoaoc.vietjack.com/thi-online/bai-tap-on-he-toan-lop-1-... |
| 1 | b630eb59da1366dd4a21748e1c0425b | Bài tập ôn hè Toán lớp 1 Chuyển đề 2: Phép cộng,... | [{ "choices": ["A. 92", "B. 38", "C. 34", "D. 93"], "explanation": "Hướng dẫn giải \n Đáp án..." | https://khoaoc.vietjack.com/thi-online/bai-tap-on-he-toan-lop-1-... |
| 1 | 6b8ee5d69959f81b1d29c79a2b173e6e | Bài tập ôn hè Toán lớp 1 Chuyển đề 1: Đếm, đọc,... | [{ "choices": ["A. hai mươi năm", "B. hai mươi lăm", "C. hai năm", "D. hai lăm"], "explanation": "..." | https://khoaoc.vietjack.com/thi-online/bai-tap-on-he-toan-lop-1-... |
| 1 | 3d201ee07af3323ef17df399fb2fa4ab | Bài tập ôn hè Toán lớp 1 Chuyển đề 4: Hình phẳng... | [{ "choices": [], "explanation": "Hướng dẫn giải \n Đáp án đúng là: A \n Đồ vật A có dạng hình trô..." | https://khoaoc.vietjack.com/thi-online/bai-tap-on-he-toan-lop-1-... |
| 1 | 0ed360e4f340d35e0e816cd8e37866514 | Đề kiểm tra giữa học kì 1 Toán lớp 1 có đáp án... | [{ "choices": [], "explanation": "Lời giải: \n a. $2 + 3 = 5$ \n b. $3 + 3 = 6$ \n c. $1 + 4 = 5$ \n d. $2 + 4 = 6$..." | https://khoaoc.vietjack.com/thi-online/de-kiem-tra-giua-hoc-ki-1-... |
| 1 | 46a2dd839aab78574f83fc9138de337b | Đề thi giữa học kì 2 môn Toán lớp 1 có đáp án ... | [{ "choices": ["A. 12", "B. 52", "C. 11", "D. 97"], "explanation": "Chọn đáp án A.", "..." | https://khoaoc.vietjack.com/thi-online/de-thi-giua-hoc-ki-2-mon-... |

Hình 14: Tổng quan về bộ dữ liệu toán cấp 1 trên HuggingFace

Như vậy, để giúp Chatbot cải thiện khả năng giải toán của mình, chúng ta sẽ thực hiện các bước code theo hướng dẫn sau đây với một file colab mới. Lưu ý rằng, có một số đoạn code sẽ tương tự như phần B, để tránh gây dài, người viết sẽ chỉ ghi lại code và không chú thích gì thêm.

1. Cài đặt thư viện:

```
1 !pip install -q -U bitsandbytes
2 !pip install -q -U datasets
3 !pip install -q -U git+https://github.com/huggingface/transformers.git
4 !pip install -q -U git+https://github.com/huggingface/peft.git
5 !pip install -q -U git+https://github.com/huggingface/accelerate.git
6 !pip install -q -U loralib
7 !pip install -q -U einops
```

2. Import các thư viện cần thiết:

```
1 import json
2 import os
3 import bitsandbytes as bnb
4 import torch
5 import torch.nn as nn
6 import transformers
7
8 from pprint import pprint
9 from datasets import load_dataset
10 from huggingface_hub import notebook_login
11 from peft import (
12     LoraConfig,
13     PeftConfig,
14     PeftModel,
15     get_peft_model,
16     prepare_model_for_kbit_training
17 )
18 from transformers import (
19     AutoConfig,
20     AutoModelForCausalLM,
21     AutoTokenizer,
22     BitsAndBytesConfig
23 )
24
25 os.environ["CUDA_VISIBLE_DEVICES"] = "0"
```

3. Khởi tạo mô hình:

```
1 MODEL_NAME = "vilm/vinallama-7b-chat"
2
3 bnb_config = BitsAndBytesConfig(
4     load_in_4bit=True,
5     bnb_4bit_use_double_quant=True,
6     bnb_4bit_quant_type="nf4",
7     bnb_4bit_compute_dtype=torch.bfloat16
8 )
9
10 model = AutoModelForCausalLM.from_pretrained(
11     MODEL_NAME,
12     device_map="auto",
13     trust_remote_code=True,
14     quantization_config=bnb_config
15 )
16
17 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
18 tokenizer.pad_token = tokenizer.eos_token
19
20 model.gradient_checkpointing_enable()
21 model = prepare_model_for_kbit_training(model)
```

```

22
23 config = LoraConfig(
24     r=16,
25     lora_alpha=32,
26     target_modules=[
27         "q_proj",
28         "up_proj",
29         "o_proj",
30         "k_proj",
31         "down_proj",
32         "gate_proj",
33         "v_proj"
34     ],
35     lora_dropout=0.05,
36     bias="none",
37     task_type="CAUSAL_LM"
38 )
39
40 model = get_peft_model(model, config)

```

4. **Xây dựng bộ dữ liệu fine-tuning:** Tương tự như phần trước, chúng ta cũng sẽ xây dựng bộ dữ liệu huấn luyện chỉ liên quan đến các câu hỏi toán. Các bước làm như sau:

(a) **Tải bộ dữ liệu:**

```
1 data = load_dataset('hllj/vi_grade_school_math_mcq')
```

(b) **Xây dựng hàm tạo prompt:** Khác với hàm tạo prompt của phần B, ở đây ta sẽ điều chỉnh lại một chút cho phù hợp với dữ liệu và ngữ cảnh của bài toán:

```

1 def generate_prompt(question, choices, explanation):
2     return f"""
3 <|im_start|>system
4 Bạn là một chuyên gia về toán. Bạn sẽ nhận câu hỏi trắc nghiệm kèm theo các l
5     ựa chọn, hãy giải step by step nếu có và chọn phương án đúng.
6 <|im_start|>user
7 ### Câu hỏi:
8 {question}
9 ### Các lựa chọn:
10 {choices}
11 ### Câu trả lời:
12
13 <|im_start|>assistant
14 {explanation}
15 """.strip()
16
17 def generate_and_tokenize_prompt(question, choices, explanation):
18     full_prompt = generate_prompt(question, choices, explanation)
19     tokenized_full_prompt = tokenizer(
20         full_prompt,
21         padding=True,
22         truncation=True
23     )
24
25     return tokenized_full_prompt

```

Tại đây, có một số điều chỉnh mà các bạn cần lưu ý:

- **Tham số đầu vào:** Hàm nhận đầu vào là 3 tham số dạng string lần lượt đại diện cho: câu hỏi của người dùng (question), các đáp án trắc nghiệm (choices) và phương án đúng kèm lời giải thích nếu có (explanation).
- **Prompt:** Cấu trúc của câu prompt có phần khác đi để phù hợp với ngữ cảnh bài toán.

```
<|im_start|>system
Bạn là một chuyên gia về toán. Bạn sẽ nhận câu hỏi trắc nghiệm kèm theo các lựa chọn, hãy giải step by step nếu có và chọn phương án đúng.

<|im_start|>user
### Câu hỏi:
{question}
### Các lựa chọn:
{choices}
### Câu trả lời:

<|im_start|>assistant
{explanation}
```

Hình 15: Format prompt cho dữ liệu giải toán

- (c) **Áp dụng hàm xây dựng prompt vào bộ dữ liệu:** Nhận thấy, thông tin trong trường samples của bộ dữ liệu là một list chứa các cặp câu hỏi và câu trả lời như hình dưới đây:

```
{'choices': ['A. 10 cm', 'B. 8 cm', 'C. 9 cm', 'D. 7 cm'],
'explanation': 'Hướng dẫn giải \n Đáp án đúng là: C \n Quan sát hình vẽ ta thấy độ dài của chiếc bút xoá là 9 cm .',
'question': 'Câu 1: \n \n Độ dài của chiếc bút xoá là:'},
{'choices': ['A. Nam', 'B. Minh', 'C. Mai', 'D. An'],
'explanation': 'Hướng dẫn giải \n Đáp án đúng là: C \n So sánh chiều cao của 4 bạn, ta thấy: 88 < 89 < 92 < 98 . \n Trong 4 số trên số nhỏ nhất là 88, tương ứng với chiều cao của bạn Mai . \n Vậy bạn thấp nhất là bạn Mai .',
'question': 'Câu 2: \n \n Nam cao 98 cm, Minh cao 92 cm, Mai cao 88 cm và An cao 89 cm. Bạn thấp nhất là:'},
{'choices': ['A. 10 cm', 'B. 12 cm', 'C. 14 cm', 'D. 15 cm'],
'explanation': 'Hướng dẫn giải \n Đáp án đúng là: D \n Chiều cao của bạn thỏ bằng chiều cao của 3 củ cà rốt . \n Chiều cao của 1 củ cà rốt là 5 cm . \n Chiều cao của bạn thỏ là: \n 5 + 5 + 5 = 15 (cm) \n Đáp số: 15 cm.',
'question': 'Câu 3: \n \n Chiều cao của bạn thỏ là:'},
{'choices': ['A. 7 giờ sáng',
'B. 7 giờ tối',
'C. 8 giờ sáng',
'D. 8 giờ tối'],
'explanation': 'Hướng dẫn giải \n Đáp án đúng là: A \n Ta có: 6 giờ sáng + 1 giờ = 7 giờ sáng . \n Vậy An đến trường lúc 7 giờ sáng .',
'question': 'Câu 4: \n \n An đi từ nhà đến trường hết 1 giờ. An bắt đầu đi từ nhà đến trường lúc 6 giờ sáng và đến trường lúc:'},
{'choices': ['A. 6 giờ', 'B. 7 giờ', 'C. 8 giờ', 'D. 9 giờ'],
'explanation': 'Hướng dẫn giải \n Đáp án đúng là: A \n Ta có: 7 giờ - 1 giờ = 6 giờ \n Vậy An đến lớp lúc 6 giờ .',
'question': 'Câu 5: \n \n Giờ vào học buổi sáng là 7 giờ. An đến sớm hơn 1 giờ. An đến lớp lúc:'}]
```

Hình 16: Nội dung trường 'problems' của một mẫu dữ liệu

Vì lẽ đó, ta sẽ truy cập vào trường thông tin này để bóc tách ra từng cặp câu hỏi - câu trả lời và áp dụng hàm tạo prompt vừa triển khai để có một mẫu dữ liệu huấn luyện hoàn chỉnh:

```
1 training_samples = []
2 for sample in tqdm(data['train']):
3     for quest in sample['problems']:
4         choices = quest['choices']
5         explanation = quest['explanation'].strip()
6         question = quest['question']
7
8         if explanation == '' or question == '' or choices == []:
9             continue
10
11        try:
12            question = question.split('\n \n')[1].strip()
13        except:
14            continue
15
```

```

16     choices = '\n'.join(choices)
17     training_sample = generate_and_tokenize_prompt(
18         question, choices, explanation
19     )
20
21     training_samples.append(training_sample)
22
23 choices_data = Dataset.from_list(training_samples)

```

Trong đó:

- **Dòng 1:** Khởi tạo một list rỗng, dùng để lưu các mẫu dữ liệu huấn luyện.
- **Dòng 2:** Duyệt qua từng hàng trong bộ dữ liệu gốc.
- **Dòng 3:** Trong trường thông tin 'problems' tại hàng đang xét, duyệt qua từng câu hỏi.
- **Từ dòng 4 đến 16:** Tách thông tin câu hỏi (question), các lựa chọn trắc nghiệm (choices) và câu trả lời (explanation).
- **Từ dòng 17 đến 23:** Gọi hàm tạo prompt để lấy mẫu dữ liệu huấn luyện hoàn chỉnh. Sau đó, đưa vào list rỗng khởi tạo ở dòng 1. Kết thúc vòng lặp, thực hiện chuyển đổi list sang HuggingFace Dataset.

5. **Thực hiện huấn luyện:** Với bộ dữ liệu mới đã chuẩn bị, ta tiến hành fine-tuning mô hình. Lưu ý rằng, thời gian huấn luyện theo đúng như mô tả bài viết sẽ tốn **hơn 3 tiếng**. Vì vậy, các bạn có thể cân nhắc hạ số training samples ở bước xây dựng bộ dữ liệu. Tuy nhiên, điều này cũng đồng nghĩa rằng độ chính xác của mô hình được fine-tuning sẽ giảm xuống.

```

1 training_args = transformers.TrainingArguments(
2     per_device_train_batch_size=1,
3     gradient_accumulation_steps=4,
4     num_train_epochs=1,
5     learning_rate=2e-4,
6     fp16=True,
7     save_total_limit=3,
8     logging_steps=1,
9     output_dir="experiments",
10    optim="paged_adamw_8bit",
11    lr_scheduler_type="cosine",
12    warmup_ratio=0.05,
13 )
14
15 trainer = transformers.Trainer(
16     model=model,
17     train_dataset=choices_data,
18     args=training_args,
19     data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=
    False)
20 )
21 model.config.use_cache = False
22 trainer.train()

```

Để kiểm tra xem mô hình sau khi qua fine-tuning có cải thiện kết quả trả về với câu hỏi toán hay không, ta sẽ sử dụng một mẫu câu hỏi đã hỏi ở lần thử đầu và lần lượt đưa cho mô hình trước và sau khi fine-tuning trả lời. Theo đó, kết quả ta nhận được lần lượt như sau:

<|im_start|> system Bạn là một chuyên gia về toán. Bạn sẽ nhận câu hỏi trắc nghiệm kèm theo các lựa chọn, hãy giải step by step nếu có và chọn phương án đúng.

<|im_start|> user
Câu hỏi: Số gồm 1 đơn vị và 2 chục đọc là :
Các lựa chọn:
A. 20
B. 21
C. 30
D. 31
Câu trả lời:

<|im_start|> assistant

Before Fine-tuning: Để giải quyết vấn đề này, chúng ta cần hiểu cách đọc số gồm 1 đơn vị và 2 chục. Một chục bằng 10, vì vậy nếu chúng ta có 2 chục, chúng ta có $2 * 10 = 20$ đơn vị. Bây giờ, chúng ta cần cộng thêm 1 đơn vị. Do đó, số đọc là $20 + 1 = 21$. Bây giờ, hãy kiểm tra các lựa chọn đã cho: A. 20 B. 21 C. 30 D. 31 Phương án đúng là D. 31 **Vậy nên, số gồm 1 đơn vị và 2 chục đọc là 31.**

After Fine-tuning: Đáp án B

Hình 17: Kết quả trước và sau khi thực hiện fine-tuning

Có thể thấy, mô hình đã cải thiện kết quả đáng kể so với trước khi được fine-tuning. Từ đây, chúng ta có thể phần nào thấy được tầm quan trọng của kỹ thuật này.

Phần III: Câu hỏi trắc nghiệm

- LLMs là viết tắt của từ gì?
 - Large Linear Modules
 - Localized Learning Mechanisms
 - Large Language Models
 - Longitudinal Learning Methods
- Chức năng chính của một mô hình ngôn ngữ lớn như ChatGPT, VinaLLaMA... là gì?
 - Lưu trữ dữ liệu người dùng
 - Tạo ra một câu văn bản phản hồi phù hợp với yêu cầu của người dùng
 - Nâng cấp hiệu quả sử dụng phần cứng của các máy chủ
 - Tạo ra một hình ảnh phản hồi phù hợp với yêu cầu của người dùng
- Kỹ thuật chính mà các mô hình ngôn ngữ lớn sử dụng là gì?
 - Thị giác máy tính (Computer Vision)
 - Khai phá luật kết hợp (Association Rule Learning)
 - Xử lý tín hiệu số (Digital Signal Processing)
 - Học sâu (Deep Learning)
- Trong LLMs, khái niệm prompt được hiểu như thế nào?
 - Một phần mềm hỗ trợ giúp tối ưu hóa hiệu suất của mô hình
 - Một kỹ thuật mã hóa thông tin riêng tư trong quá trình đào tạo mô hình
 - Một câu hỏi hoặc yêu cầu mà người dùng đưa ra cho mô hình
 - Một thuật toán đặc biệt để phân loại dữ liệu đầu vào
- Khái niệm fine-tuning trong LLMs là gì?
 - Quá trình cập nhật giao diện người dùng của mô hình
 - Quá trình điều chỉnh và tối ưu hóa mô hình dựa trên một tập dữ liệu cụ thể
 - Kỹ thuật mã hóa dữ liệu để bảo vệ thông tin người dùng
 - Quá trình làm mát thiết bị chạy mô hình để tăng hiệu suất
- Các mô hình ngôn ngữ lớn như ChatGPT được huấn luyện trên loại dữ liệu nào?
 - Chỉ văn bản từ sách và báo
 - Chỉ dữ liệu hình ảnh
 - Dữ liệu văn bản từ nhiều nguồn khác nhau trên Internet
 - Chỉ dữ liệu âm thanh và video
- Zero-shot learning trong ngữ cảnh của LLMs có ý nghĩa gì?
 - Mô hình chỉ được huấn luyện một lần và không bao giờ cập nhật
 - Khả năng của mô hình để hiểu và thực hiện nhiệm vụ mà nó không được huấn luyện trực tiếp
 - Mô hình chỉ sử dụng một dữ liệu duy nhất để học

- (d) Khả năng của mô hình để xóa bỏ các thông tin không cần thiết
8. Khi nói về LLMs, "Generalization" có nghĩa là gì?
- (a) Khả năng của mô hình để phân loại chung các loại dữ liệu
 - (b) Khả năng của mô hình để hoạt động tốt trên các loại dữ liệu chưa từng thấy
 - (c) Quá trình tạo ra các mô hình mới từ một mô hình hiện tại
 - (d) Tính năng giúp tăng cường bảo mật cho mô hình
9. LLMs thường sử dụng kiến trúc mạng nơ-ron nào?
- (a) Mạng Nơ-ron Hồi Quy (RNN)
 - (b) Mạng Nơ-ron Tích chập (CNN)
 - (c) Mạng Nơ-ron Transformer
 - (d) Mạng Nơ-ron Perceptron Đa lớp (MLP)
10. Chatbot sử dụng LLMs có thể áp dụng trong lĩnh vực nào?
- (a) Chỉ trong lĩnh vực giáo dục
 - (b) Chỉ trong lĩnh vực y tế
 - (c) Trong nhiều lĩnh vực như dịch vụ khách hàng, giáo dục, y tế, v.v.
 - (d) Chỉ trong lĩnh vực kỹ thuật và công nghệ

- *Hết* -