

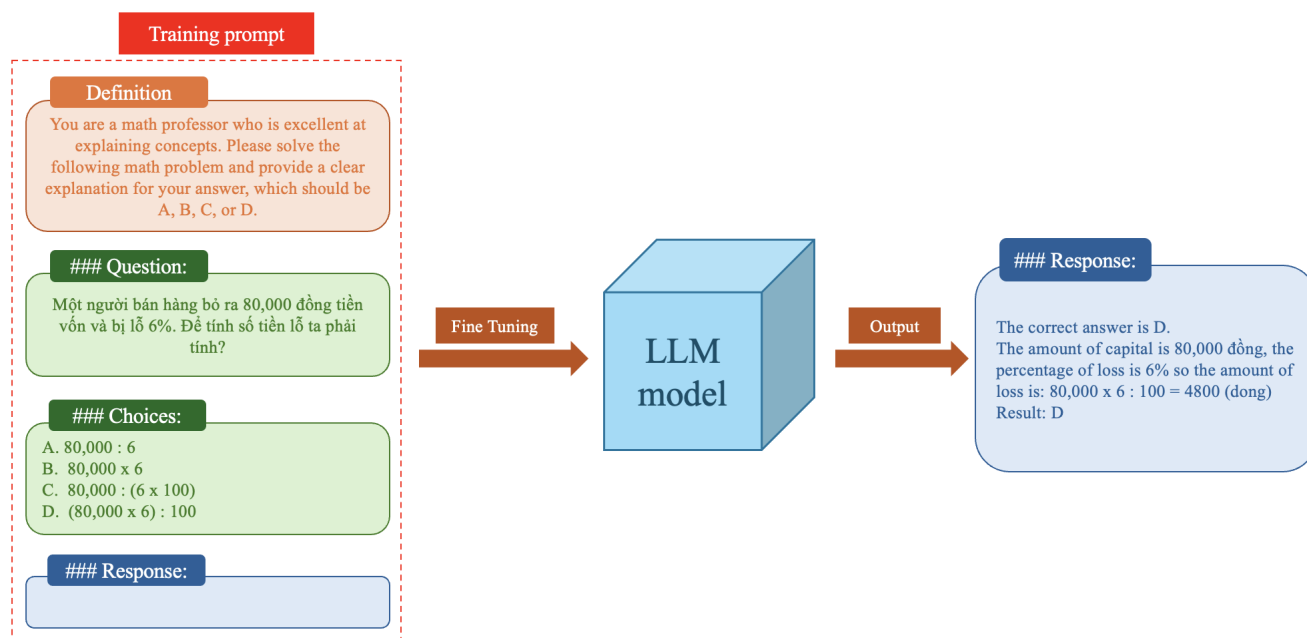
LLMs Series: Cải thiện khả năng giải toán trắc nghiệm của LLMs với Instruction Tuning

Dinh-Thang Duong, Nguyen-Thuan Duong và Quang-Vinh Dinh
PR-Team: Hoàng-Nguyên Vũ, Đăng-Nhã Nguyễn và Minh-Châu Phạm

Ngày 14 tháng 4 năm 2024

Phần I: Giới thiệu

Instruction Tuning (IT) là một trong những kỹ thuật training mô hình ngôn ngữ lớn (LLMs) rất quan trọng. Trong đó, IT giúp chúng ta cải thiện khả năng của mô hình cũng như kiểm soát kết quả đầu ra. Là kiểu huấn luyện mô hình có giám sát từ bộ dữ liệu theo cặp (instruction-output), từ đó giúp mô hình thu hẹp khoảng cách giữa từ kế tiếp được sinh ra và sự chỉ dẫn của con người.



Hình 1: Fine-tuning mô hình ngôn ngữ lớn với dữ liệu instruction giải toán trắc nghiệm

Trong bài viết này, chúng ta sẽ huấn luyện một mô hình ngôn ngữ lớn tiếng Việt với dữ liệu instruction giải toán trắc nghiệm. Từ đó, mô hình này sẽ có thể phần nào cải thiện khả năng giải toán.

Phần II: Cài đặt chương trình

Trong phần này, chúng ta sẽ xây dựng một Chatbot chuyên giải toán trắc nghiệm tiếng Việt 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:

```
<|im_start|>system
Bạn là một trợ lí AI hữu ích. Hãy trả lời người dùng một cách chính xác.
<|im_end|>
<|im_start|>user
{your_task}
<|im_end|>
<|im_start|>assistant
```

Hình 2: Format prompt của VinaLLaMA. 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()
```

Hình 3: Minh họa về cách xây dựng prompt theo format của mô hình VinaLLaMA

- (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 4: 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).

5. **Tải bộ dữ liệu fine-tuning:** Trong bài này, vì mục tiêu của chúng ta là xây dựng chatbot chuyên dùng để giải toán trắc nghiệm tiếng Việt, nên ở phần sau chúng ta sẽ thực hiện fine-tuning VinaLLaMA trên bộ dữ liệu toán trắc nghiệm để cải thiện chất lượng câu trả lời. Ta thực hiện tải bộ dữ liệu có tên là **vi_grade_school_math_mcq** như sau:

```
1 data = load_dataset('h11j/vi_grade_school_math_mcq')
```

Dataset Viewer Auto-converted to Parquet </> API View in Dataset Viewer

Split (1)
train · 2.73k rows

Search this dataset

| grade
string · classes | id
string · lengths | title
string · lengths | problems
list |
|---------------------------|----------------------------------|---|---|
| 5 values | 32 32 | 16 117 | |
| 1 | 34a7a20a1cec28e5a0275dec1c9a245e | Bài tập ôn hè Toán lớp 1 Chuyên đề 5... | [{ "choices": ["A. 10 cm", "B. 8 c", "D. 7 cm"], "explanation": "Hướng d |
| 1 | 490508bdbe190af20a2cd30153ee1f0a | Bài tập ôn hè Toán lớp 1 Chuyên đề 3... | [{ "choices": ["A. 5 quả", "B. 4 q", "D. 9 quả"], "explanation": " |
| 1 | b630eb59da13666dd4a21748e1c8425b | Bài tập ôn hè Toán lớp 1 Chuyên đề 2... | [{ "choices": ["A. 92", "B. 38", "93"], "explanation": "Hướng dẫn giả |
| 1 | 6b8ee5d69959f81b1d29c79a2b173e6e | Bài tập ôn hè Toán lớp 1 Chuyên đề 1... | [{ "choices": ["A. hai mươi năm", "lăm", "C. hai năm", "D. hai lăm"], |
| 1 | 3d201ee07af3323ef17df399fb2fa4ab | Bài tập ôn hè Toán lớp 1 Chuyên đề 4... | [{ "choices": [], "explanation": "H\n\nĐáp án đúng là: A \nỒ vật A có |
| 1 | 0ed360e4f340d356e816cd8e37866514 | Đề kiểm tra giữa | [{ "choices": [], "explanation": "L |

< Previous 1 2 3 ... 28 Next >

Hình 5: 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: ['problems', 'grade', 'id', 'title', 'url'],
    num_rows: 2733
  })
})
```

Hình 6: Dữ liệu của biến `data`. Biến có kiểu dữ liệu là **DatasetDict**, một kiểu dữ liệu riêng biệt của thư viện **datasets**

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:

- 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(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
        ựa chọn, hãy giải step by step nếu có và chọn phương án đúng.
5
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

```

- (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(question, choices, explanation):
2     full_prompt = generate_prompt(question, choices, explanation)
3     tokenized_full_prompt = tokenizer(
4         full_prompt,
5         padding=True,
6         truncation=True
7     )
8
9     return tokenized_full_prompt

```

- (c) **Áp dụng tokenization vào bộ dữ liệu:** Với hàm tokenization vừa xây dựng, ta sử dụng đoạn code sau đây để tách các thông tin về các lựa chọn trắc nghiệm (choices), lời giải thích kèm đáp án (explanation) và câu hỏi (question). Sau đó, đưa các thông tin này vào hàm tokenization để hình thành câu prompt cho mô hình. Sau đó, sử dụng hàm `Dataset.from_list()`

```

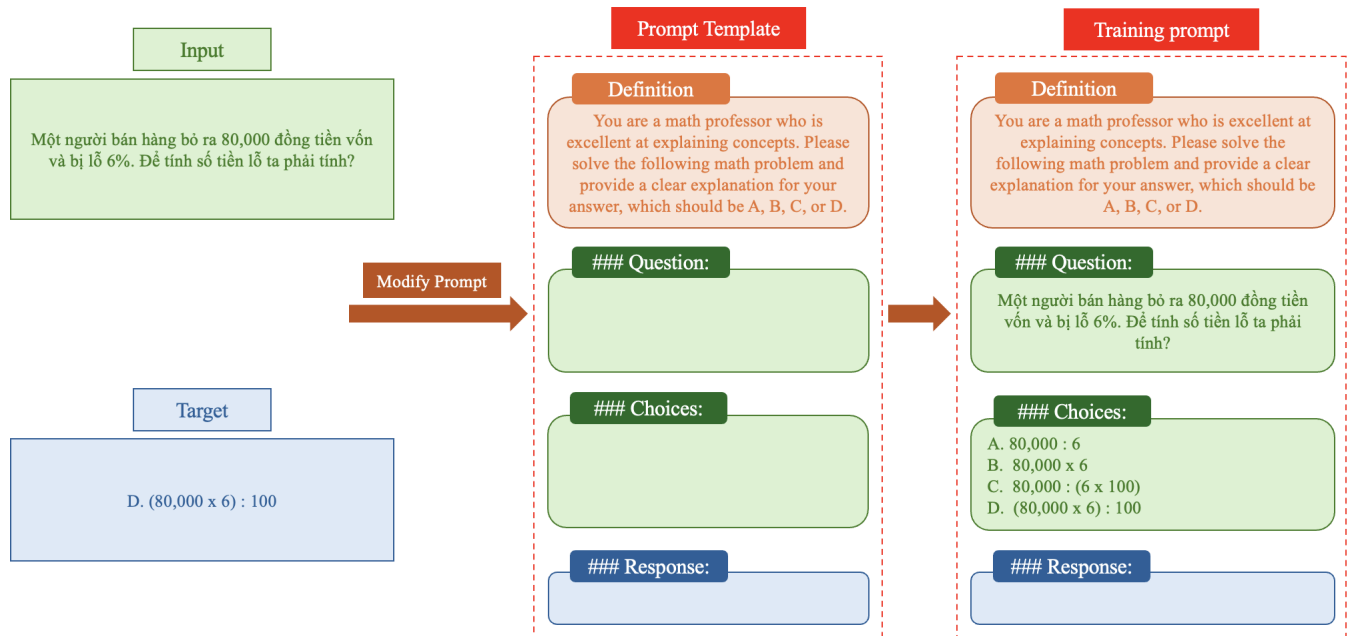
1 training_samples = []
2 for sample in tqdm(data['train']):
3     for quest in sample['problems']:
4         choices = quest['choices']
5         explanation = quest['explanation'].strip()
6         question = quest['question']
7

```

```

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)

```



Hình 7: Minh họa về mẫu dữ liệu instruction giải toán trắc nghiệm.

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 )

```

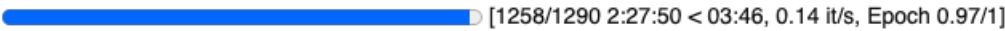


```

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.



[1258/1290 2:27:50 < 03:46, 0.14 it/s, Epoch 0.97/1]

| 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 8: 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 chuyên gia về toán. Bạn sẽ nhận câu hỏi trắc nghiệm kèm theo các lựa
7 chọn, hãy giải step by step nếu có và chọn phương án đúng.
8 <|im_start|>user
9 ### Câu hỏi:
10 1 + 1 =
11 ### Các lựa chọn:
12 A. 1
13 B. 2
14 C. 3
15 D. 4
16 ### Câu trả lời:

```

```
17
18 <|im_start|>assistant
19 """.strip()
20
21 encoding = tokenizer(prompt, return_tensors="pt").to(device)
22 with torch.inference_mode():
23     outputs = model.generate(
24         input_ids=encoding.input_ids,
25         attention_mask=encoding.attention_mask,
26         generation_config=generation_config
27     )
28
29 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 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:
1 + 1 =
### Các lựa chọn:
A. 1
B. 2
C. 3
D. 4
### Câu trả lời:

<|im_start|> assistant
Đáp án B 1 + 1 = 2
Chọn B. \
```

Hình 9: Câu trả lời của mô hình

Phần III: Câu hỏi trắc nghiệm

1. Trong ngữ cảnh về mô hình ngôn ngữ lớn (LLMs), Instruction Tuning được hiểu như thế nào?
 - (a) Huấn luyện mô hình trên task mới mà không cần mẫu dữ liệu nào.
 - (b) Điều chỉnh kết quả của mô hình trong quá trình deploy.
 - (c) Huấn luyện mô hình để tuân theo các yêu cầu cụ thể (instruction).
 - (d) Giảm kích thước của mô hình để tăng độ hiệu quả.
2. Instruction tuning là một dạng của kiểu học
 - (a) Supervised learning.
 - (b) Self-supervised learning.
 - (c) Unsupervised learning.
 - (d) Reinforcement learning.
3. Khi thực hiện instruction tuning, hàm loss nào sau đây có thể được sử dụng?
 - (a) Mean Squared Error (MSE)
 - (b) Hinge Loss
 - (c) Kullback-Leibler Divergence
 - (d) Cross-Entropy Loss
4. Mệnh đề sau đúng hay sai: "Ta luôn nên áp dụng instruction tuning để giảm thiểu chi phí tính toán trong quá trình training"?
 - (a) Đúng
 - (b) Sai
5. Trong LLMs, khái niệm prompt được hiểu như thế nào?
 - (a) Một phần mềm hỗ trợ giúp tối ưu hóa hiệu suất của mô hình
 - (b) Một kỹ thuật mã hóa thông tin riêng tư trong quá trình đào tạo mô hình
 - (c) Một câu hỏi hoặc yêu cầu mà người dùng đưa ra cho mô hình
 - (d) Một thuật toán đặc biệt để phân loại dữ liệu đầu vào
6. Trong LLMs, ta nên áp dụng kỹ thuật nào sau đây để cải thiện khả năng thực hiện một task cụ thể nào đó của mô hình mà không cần training?
 - (a) Sử dụng kỹ thuật transfer learning.
 - (b) Ứng dụng khả năng zero-shot learning của mô hình.
 - (c) Lập trình thủ công tại bước hậu xử lý cho mỗi task.
 - (d) Mở rộng bộ dữ liệu training.
7. Kỹ thuật prompting nào dưới đây cung cấp cho mô hình chỉ một ví dụ về task cần làm?
 - (a) One-shot learning
 - (b) Few-shot learning
 - (c) Continuous learning
 - (d) Transfer learning

8. Câu nào sau đây mô tả đúng về kỹ thuật Parameter Efficient Fine-tuning (PEFT)?
- (a) Một kỹ thuật dùng để huấn luyện mô hình trên bộ dữ liệu cực lớn.
 - (b) Một kỹ thuật liên quan đến việc cập nhật một phần nhỏ tham số của mô hình khi huấn luyện.
 - (c) Một kỹ thuật huấn luyện dành riêng cho các mô hình có kích thước nhỏ (dưới 1 tỷ tham số).
 - (d) Một kỹ thuật để tăng chi phí tính toán của mô hình.
9. Mệnh đề sau đúng hay sai: "Low-Rank Adaptation (LoRA) là một kỹ thuật về PEFT"?
- (a) Đúng.
 - (b) Sai.
10. So với LoRA, QLoRA có điểm gì khác biệt gì trong việc huấn luyện LLMs?
- (a) QLoRA lượng tử hóa (quantize) tham số mô hình; LoRA thì không.
 - (b) QLoRA sử dụng nhiều tham số mô hình; LoRA ít hơn.
 - (c) QLoRA tối ưu khả năng tổng quát của mô hình; LoRA tối ưu trên một task cụ thể.
 - (d) QLoRA giảm kích thước mô hình; LoRA tăng lên.

- Hết -