

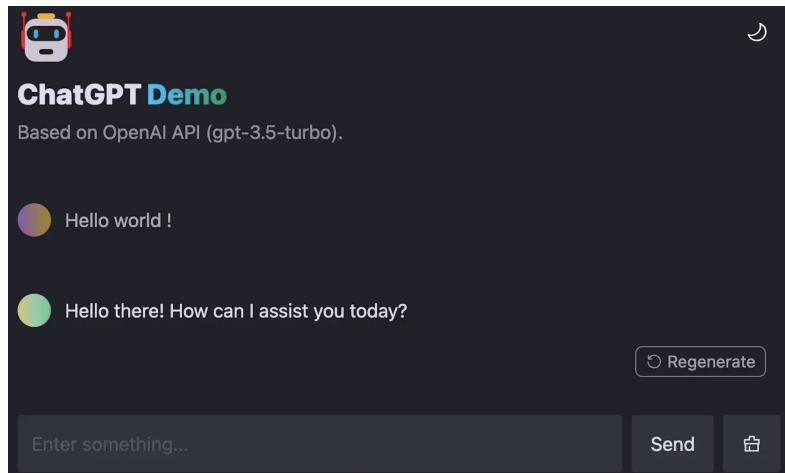
Project: Reinforcement Learning From Human Feedback

Quoc-Thai Nguyen và Quang-Vinh Dinh

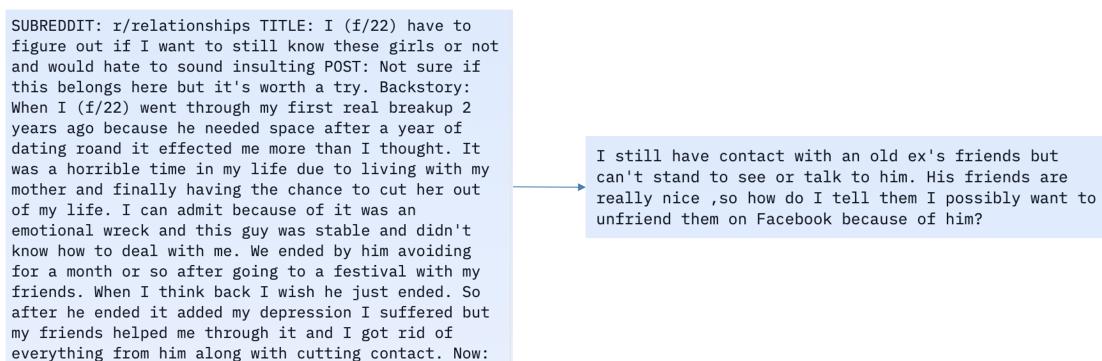
PR-Team: Dăng-Nhã Nguyễn, Minh-Châu Phạm và Hoàng-Nguyễn Vũ

Ngày 1 tháng 4 năm 2024

Phần I. Giới thiệu



Hình 1: Ứng dụng ChatGPT được cải tiến dựa vào mô hình học tăng cường từ phản hồi người dùng.



Hình 2: Ví dụ minh họa tóm tắt văn bản sử dụng mô hình học tăng cường từ phản hồi người dùng.

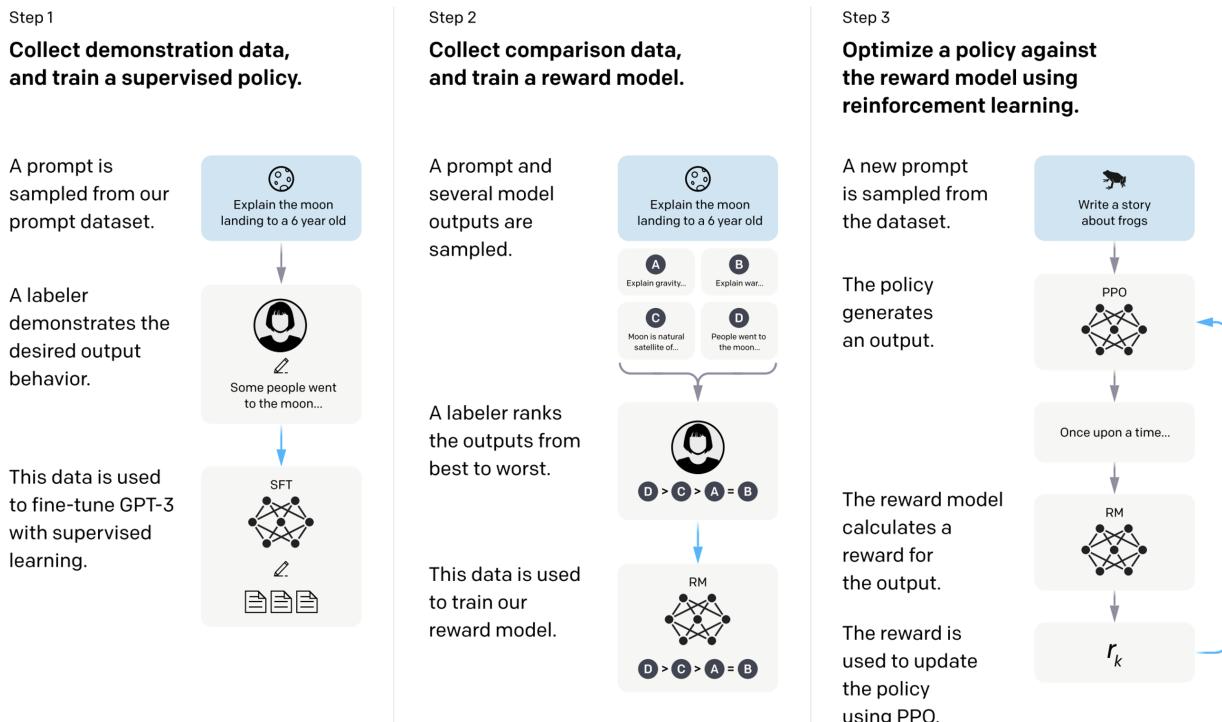
ChatGPT (Chat Generative Pre-Training Transformer) là mô hình chatbot được phát triển bởi OpenAI ra mắt vào tháng 11 năm 2022. ChatGPT được xây dựng dựa trên việc tinh chỉnh mô hình ngôn ngữ lớn (Large Language Model) kết hợp với học tăng cường dựa trên phản hồi của con người (Reinforcement Learning with Human Feedback).

ChatGPT có thể giải quyết nhiều bài toán khác nhau trong xử lý tự nhiên như: phân loại văn bản (Text Classification), bài toán dịch máy (Machine Translation), bài toán hỏi đáp (Question Answering), bài toán tóm tắt văn bản (Text Summarization),... Từ các bài toán gốc cơ bản này, ChatGPT được triển khai cho nhiều lĩnh vực khác nhau: giáo dục, sáng tạo nội dung, chăm sóc sức khoẻ,...

Mô hình ChatGPT còn có khả năng tương thích, tối ưu cho bài toán cụ thể trong quá trình dự đoán (Inference - đưa ra phản hồi cho các bài toán) thông qua phương pháp học tập trong ngữ cảnh (In-Content Learning). Phương pháp học tập trong ngữ cảnh được xây dựng thông qua các **prompt**, được hiểu là các dữ liệu đầu vào với mục tiêu hướng dẫn hoặc mô tả chi tiết cho bài toán giúp ChatGPT có thể xác định được bài toán cần xử lý và tối ưu miền xử lý, nhờ đó ChatGPT có thể cho kết quả tốt hơn.

- (a) Zero-shot: Dựa ra câu trả lời chỉ dựa vào mô tả của bài toán
- (b) One-shot: Dựa ra câu trả lời dựa vào mô tả bài toán và một ví dụ mẫu cho bài toán
- (c) One-shot: Dựa ra câu trả lời dựa vào mô tả bài toán và một vài ví dụ mẫu cho bài toán

Một trong những kỹ thuật giúp cho mô hình ngôn ngữ đạt kết quả tốt là cải tiến các mô hình này dựa vào học tăng cường từ phản hồi của người dùng (RLHF - Reinforcement learning from human feedback). Trong phần này chúng ta sẽ tập trung vào đi sâu tìm hiểu về RLHF.



Hình 3: Reinforcement Learning from Human Feedback.

(a) Supervised fine-tuning (SFT)

Trong phần này chúng ta sử dụng 1 tập các prompt và response để tinh chỉnh mô hình ngôn ngữ. Mô hình GPT2 được chọn với 12 khối Transformer-Decoder. Ngoài ra, có nhiều mô hình ngôn ngữ có thể chọn phù hợp với mục tiêu và open source như BLOOM, LLaMa,...

(b) Reward Modeling (RM)

Trong phần này chúng ta sử dụng tập dữ liệu Helpful and Harmless để huấn luyện mô hình RM. Với mỗi prompt chúng ta sẽ có 1 response là có nhãn "chosen" và 1 response có nhãn là "rejected". Mục tiêu của phần này với mỗi prompt và response chúng ta cần đưa ra điểm số tương ứng.

(c) Reinforcement learning (RL)

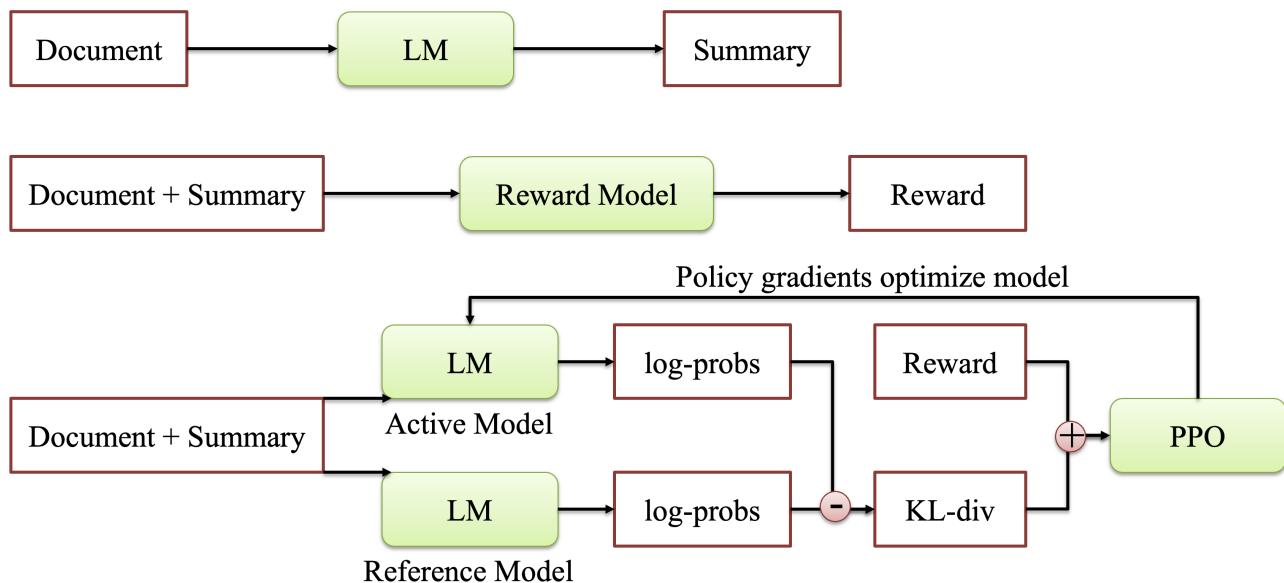
Trong phần này chúng ta sử dụng bộ dữ liệu rm-static, kết hợp với mô hình RM để tinh chỉnh mô hình SFT dựa vào phương pháp huấn luyện sử dụng Proximal Policy Optimization (PPO).

Phần II. Text Summarization using RLHF

Trong phần này, chúng ta sẽ tập trung ứng dụng cải tiến mô hình dựa vào RLHF cho bài toán tóm tắt văn bản.

Tóm tắt văn bản (Text Summarization) nhận đầu vào là đoạn văn bản dài hoặc nhiều đoạn văn bản dài. Thông qua mô hình các đoạn văn bản sẽ được tóm tắt ngắn gọn thành vài câu. Ví dụ minh họa về bài toán tóm tắt văn bản được minh họa như hình 2.

Các bước để huấn luyện và tối ưu mô hình được mô tả như sau:



Hình 4: Các bước huấn luyện mô hình tóm tắt văn bản sử dụng RLHF.

RLHF bao gồm 3 phần:

- Supervised Fine-Tuning: Tinh chỉnh mô hình trên bộ dữ liệu có nhãn cho bài toán tóm tắt
- Reward Modeling: Huấn luyện mô hình cho điểm chất lượng bản tóm tắt
- Reinforcement Learning: Tinh chỉnh mô hình dựa vào học tăng cường để cải tiến mô hình

1. Supervised Fine-Tuning (SFT)

Trong phần này chúng ta sẽ tinh chỉnh mô hình tiền huấn luyện như GPT, BLOOM, FlanT5,... trên bộ dữ liệu được gán nhãn. Bộ dữ liệu `openai_summarize_tldr` bao gồm các cặp dữ liệu: văn bản và bản tóm tắt.



SUBREDDIT: r/relationships TITLE: I (f/22) have to figure out if I want to still know these girls or not and would hate to sound insulting POST: Not sure if this belongs here but it's worth a try. Backstory: When I (f/22) went through my first real breakup 2 years ago because he needed space after a year of dating and it effected me more than I thought. It was a horrible time in my life due to living with my mother and finally having the chance to cut her out

I still have contact with an old ex's friends but can't stand to see or talk to him. His friends are really nice, so how do I tell them I possibly want to unfriend them on Facebook because of him?

Hình 5: Supervised Fine-Tuning.

1.1. Dataset

Chúng ta tải về bộ dữ liệu từ thư viện transformers sau đó tiến hành xây dựng cấu trúc dữ liệu như sau: "Text: document # Summary: summary"

```

1 # install libs
2 !pip install -q datasets==2.18.0 trl==0.8.1 evaluate==0.4.1 rouge_score==0.1.2 peft
3 ==0.10.0
4
5 # load the dataset
6 from datasets import load_dataset
7
8 sft_ds_name = 'CarperAI/openai_summarize_tldr'
9 sft_ds = load_dataset(sft_ds_name)
10 sft_train = sft_ds['train']
11 sft_valid = sft_ds['valid']
12 sft_test = sft_ds['test']
13
14 def formatting_func(example):
15     text = f"### Text: {example['prompt']}\n### Summary: {example['label']}"
16     return text
17
18 # demo formating
19 for example in sft_train:
20     print(formatting_func(example))
      break
  
```

1.2. Model

Mô hình tiền huấn luyện được sử dụng là OPT. Để tăng tốc độ huấn luyện mô hình, chúng ta sẽ sử dụng kỹ thuật quantization và LORA.

```

1 import torch
2 from trl import ModelConfig, get_quantization_config, get_kbit_device_map
3 from peft import LoraConfig, PeftConfig, PeftModel, get_peft_model,
4   prepare_model_for_kbit_training
  
```

```

4
5 model_config = ModelConfig(
6     model_name_or_path='facebook/opt-350m'
7 )
8
9 torch_dtype = (
10     model_config.torch_dtype
11     if model_config.torch_dtype in ["auto", None]
12     else getattr(torch, model_config.torch_dtype)
13 )
14 quantization_config = get_quantization_config(model_config)
15 model_kwargs = dict(
16     revision=model_config.model_revision,
17     trust_remote_code=model_config.trust_remote_code,
18     attn_implementation=model_config.attn_implementation,
19     torch_dtype=torch_dtype,
20     use_cache=False,
21     device_map=get_kbit_device_map() if quantization_config is not None else None,
22     quantization_config=quantization_config,
23 )
24 tokenizer = AutoTokenizer.from_pretrained(model_config.model_name_or_path, use_fast=
25     True)
25 tokenizer.pad_token = tokenizer.eos_token
26
27 tokenizer.pad_token_id = tokenizer.eos_token_id
28
29
30 # lora
31 peft_config = LoraConfig(
32     r=16,
33     lora_alpha=32,
34     lora_dropout=0.05,
35     bias="none",
36     task_type="CAUSAL_LM",
37 )

```

1.3. Metric

Chúng ta sử dụng độ ROUGE để đánh giá mô hình

```

1 import evaluate
2
3 rouge = evaluate.load("rouge")
4
5 def compute_metrics(eval_preds):
6     if isinstance(eval_preds, tuple):
7         eval_preds = eval_preds[0]
8     labels_ids = eval_preds.label_ids
9     pred_ids = eval_preds.predictions
10    pred_str = tokenizer.batch_decode(pred_ids, skip_special_tokens=True)
11    label_str = tokenizer.batch_decode(labels_ids, skip_special_tokens=True)
12    result = rouge.compute(predictions=pred_str, references=label_str)
13    return result

```

1.4. Trainer

Chúng ta sẽ thiết lập các tham số huấn luyện mô hình

```

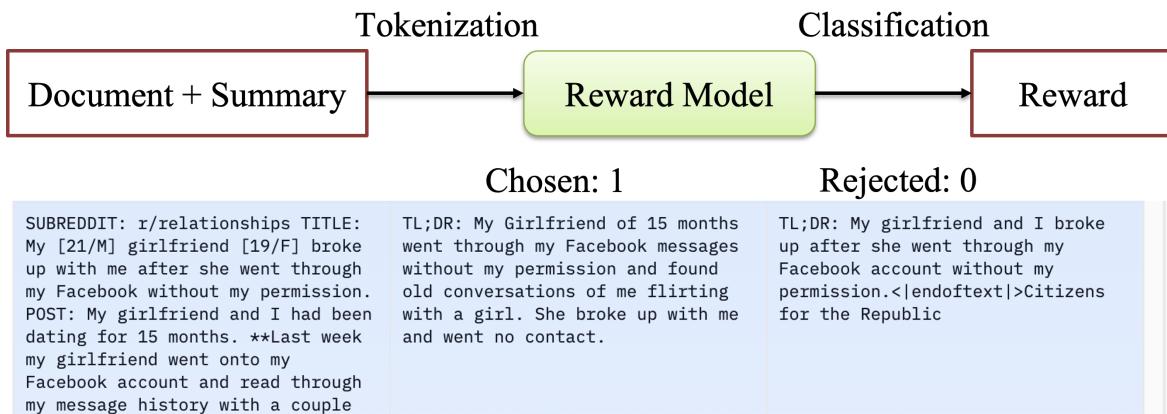
1 from trl import SFTTrainer
2 from transformers import TrainingArguments
3
4 num_epochs = 10
5 training_args = TrainingArguments(
6     output_dir='./save_model',

```

```
7     evaluation_strategy="epoch",
8     save_strategy='epoch',
9     per_device_train_batch_size=4,
10    per_device_eval_batch_size=4,
11    adam_beta1=0.9,
12    adam_beta2=0.95,
13    num_train_epochs=num_epochs,
14    load_best_model_at_end=True,
15 )
16
17 max_input_length = 512
18 trainer = SFTTrainer(
19     model=model_config.model_name_or_path,
20     model_init_kwargs=model_kwargs,
21     args=training_args,
22     train_dataset=sft_train,
23     eval_dataset=sft_valid,
24     max_seq_length=max_input_length,
25     tokenizer=tokenizer,
26     peft_config=peft_config,
27     compute_metrics=compute_metrics,
28     packing=True,
29     formatting_func=formatting_func
30 )
31 trainer.train()
```

2. Reward Modeling

Trong phần này chúng ta sẽ xây dựng mô hình đánh giá chất lượng bản tóm tắt. Đầu vào của mô hình là văn bản kết hợp với bản tóm tắt. Đầu ra là giá trị dự đoán từ 0 đến 1. Chúng ta sẽ tinh chỉnh mô hình SFT cho bài toán phân loại. Với bản tóm tắt phù hợp (Chosen) sẽ có nhãn là 1, và bản tóm tắt không phù hợp (Rejected) sẽ có nhãn là 0.



Hình 6: Reward Modeling.

2.1. Dataset

Bộ dữ liệu `openai_summarize_comparisons` bao gồm 83 mẫu có nhãn: văn bản, bản tóm tắt phù hợp và bản tóm tắt không phù hợp. Chúng ta sẽ kết hợp lần lượt văn bản với bản tóm tắt phù hợp và không phù hợp để huấn luyện mô hình.

```

1 # load dataset
2 from datasets import load_dataset
3
4 rw_ds_name = 'CarperAI/openai_summarize_comparisons'
5 rw_ds = load_dataset(rw_ds_name)
6
7 def preprocess_function(examples):
8     new_examples = {
9         "input_ids_chosen": [],
10        "attention_mask_chosen": [],
11        "input_ids_rejected": [],
12        "attention_mask_rejected": []
13    }
14    for prompt, chosen, rejected in zip(examples["prompt"], examples["chosen"], examples["rejected"]):
15        chosen = f"### Text: {prompt}\n### Summary: {chosen}"
16        tokenized_chosen = tokenizer(chosen)
17
18        rejected = f"### Text: {prompt}\n### Summary: {rejected}"
19        tokenized_rejected = tokenizer(rejected)
20
21        new_examples["input_ids_chosen"].append(tokenized_chosen["input_ids"])
22        new_examples["attention_mask_chosen"].append(tokenized_chosen["attention_mask"])
23    ]
24        new_examples["input_ids_rejected"].append(tokenized_rejected["input_ids"])
25        new_examples["attention_mask_rejected"].append(tokenized_rejected["attention_mask"])
26
27    return new_examples
28 rw_ds_processed = rw_ds.map(

```

```

29     preprocess_function,
30     batched=True,
31     num_proc=4,
32 )
33
34 max_input_length = 512
35
36 rw_ds_filted = rw_ds_processed.filter(
37     lambda x: len(x["input_ids_chosen"]) <= max_input_length
38     and len(x["input_ids_rejected"]) <= max_input_length
39 )
40
41 rw_train = rw_ds_filted["train"]
42 rw_valid = rw_ds_filted["valid1"]
43 rw_test = rw_ds_filted["test"]

```

2.2. Model

Mô hình được sử dụng sẽ là mô hình SFT và tinh chỉnh để phù hợp cho bài toán phân loại mức câu.

```

1 import torch
2 from trl import ModelConfig, get_quantization_config, get_kbit_device_map
3 from transformers import AutoModelForSequenceClassification
4
5 model_config = ModelConfig(
6     model_name_or_path='facebook/opt-350m' # ./save_sft_model/checkpoint-1000
7 )
8
9 torch_dtype = (
10     model_config.torch_dtype
11     if model_config.torch_dtype in ["auto", None]
12     else getattr(torch, model_config.torch_dtype)
13 )
14 quantization_config = get_quantization_config(model_config)
15 model_kwarg = dict(
16     revision=model_config.model_revision,
17     trust_remote_code=model_config.trust_remote_code,
18     attn_implementation=model_config.attn_implementation,
19     torch_dtype=torch_dtype,
20     use_cache=False,
21     device_map=get_kbit_device_map() if quantization_config is not None else None,
22     quantization_config=quantization_config,
23 )
24 tokenizer = AutoTokenizer.from_pretrained(model_config.model_name_or_path, use_fast=True)
25 model = AutoModelForSequenceClassification.from_pretrained(
26     model_config.model_name_or_path, num_labels=1, **model_kwarg
27 )
28
29 peft_config = LoraConfig(
30     r=16,
31     lora_alpha=32,
32     lora_dropout=0.05,
33     bias="none",
34     task_type="SEQ_CLS",
35 )

```

2.3. Trainer

Huấn luyện mô hình Reward

```

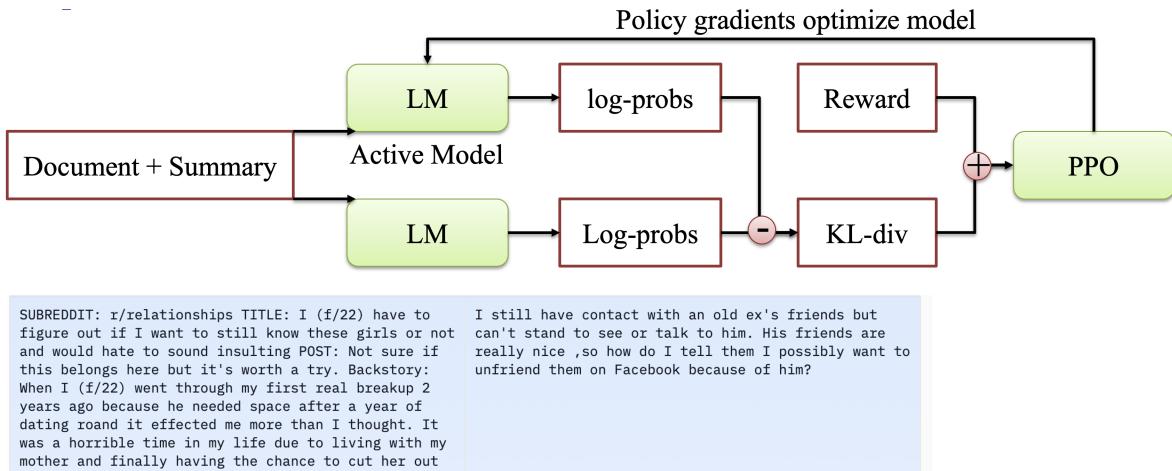
1 from trl import RewardConfig
2 from trl import RewardTrainer
3

```

```
4 num_epochs = 1 # 10
5
6 reward_config = RewardConfig(
7     output_dir='./save_rw_model',
8     evaluation_strategy="epoch",
9     save_strategy='epoch',
10    per_device_train_batch_size=4,
11    per_device_eval_batch_size=4,
12    num_train_epochs=num_epochs,
13    load_best_model_at_end=True,
14    max_length=max_input_length,
15 )
16
17 trainer = RewardTrainer(
18     model=model,
19     tokenizer=tokenizer,
20     args=reward_config,
21     train_dataset=rw_train,
22     eval_dataset=rw_valid,
23     peft_config=peft_config,
24 )
25
26 trainer.train()
```

3. Reinforcement Learning

Trong phần này chúng ta tối ưu mô hình SFT dựa vào mô hình Reward sử dụng thuật toán Proximal Policy Optimization (PPO).



Hình 7: Proximal Policy Optimization.

2.1. Dataset

Chúng ta sử dụng bộ dữ liệu `openai_summarize_tldr`. Nối phần văn bản và phần bản tóm tắt để huấn luyện mô hình.

```

1 from transformers import AutoTokenizer
2
3 ppo_ds_name = 'CarperAI/openai_summarize_tldr'
4 ppo_ds = load_dataset(sft_ds_name, split="train")
5
6 def build_dataset(ds, tokenizer, max_length=200):
7     ds = ds.filter(lambda x: len(x["prompt"]) > max_length, batched=False)
8
9     def tokenize(sample):
10         sample["text"] = sample["prompt"] + sample["label"]
11         sample["input_ids"] = tokenizer.encode(sample["text"])[0: max_length]
12         sample["query"] = tokenizer.decode(sample["input_ids"])
13         return sample
14
15     ds = ds.map(tokenize, batched=False)
16     ds.set_format(type="torch")
17     return ds
18
19 tokenizer = AutoTokenizer.from_pretrained("facebook/opt-350m")
20 tokenizer.pad_token = tokenizer.eos_token
21 ppo_ds = build_dataset(ppo_ds, tokenizer)

```

2.2. Model

Chúng ta sử dụng mô hình CausalLMWithValueHead để khởi tạo mô hình

```

1 from trl import AutoModelForCausalLMWithValueHead
2
3 from peft import LoraConfig
4
5 peft_config = LoraConfig(
6     r=16,
7     lora_alpha=32,
8     lora_dropout=0.05,

```

```

9     bias="none",
10    task_type="SEQ_CLS",
11 )
12
13 model_path = "./save_sft_model/checkpoint-1000"
14 model = AutoModelForCausalLMWithValueHead.from_pretrained(
15     pretrained_model_name_or_path=model_path,
16     peft_config=peft_config,
17 )

```

2.3. Trainer

```

1 from trl import PPOConfig, PPOTrainer
2
3 def collator(data):
4     return {key: [d[key] for d in data] for key in data[0]}
5
6 ppo_config = PPOConfig(
7     model_name="facebook/opt-350m"
8 )
9
10 device = 0 if torch.cuda.is_available() else "cpu"
11
12 ppo_trainer = PPOTrainer(ppo_config, model, None, tokenizer, dataset=ppo_ds,
13                          data_collator=collator)

```

2.4. Reward Model

Load lại mô hình reward được huấn luyện ở bước 2

```

1 from transformers import AutoModelForSequenceClassification, pipeline
2
3 rw_model = model = AutoModelForSequenceClassification.from_pretrained('./save_rw_model')
4 sentiment_pipe = pipeline("sentiment-analysis", model=rw_model, device=device)
5
6 if sentiment_pipe.tokenizer.pad_token_id is None:
7     sentiment_pipe.tokenizer.pad_token_id = tokenizer.pad_token_id
8
9 if sentiment_pipe.model.config.pad_token_id is None:
10    sentiment_pipe.model.config.pad_token_id = tokenizer.pad_token_id

```

2.5. Training

Huấn luyện mô hình với các tham số cài đặt cho quá trình sinh bản tóm tắt và đánh giá bản tóm tắt.

```

1 from tqdm import tqdm
2
3 generation_kwargs = {
4     "min_length": -1,
5     "top_k": 0.0,
6     "top_p": 1.0,
7     "do_sample": True,
8     "pad_token_id": tokenizer.eos_token_id,
9     "max_new_tokens": 200,
10 }
11 sent_kwargs = {"return_all_scores": True, "function_to_apply": "none", "batch_size": 16}
12
13 for epoch, batch in tqdm(enumerate(ppo_trainer.dataloader)):
14     query_tensors = batch["input_ids"]
15
16     # Get response from gpt2

```

```
17     response_tensors, ref_response_tensors = ppo_trainer.generate(
18         query_tensors, return_prompt=False, generate_ref_response=True, **
19         generation_kwargs
20     )
21     batch["response"] = tokenizer.batch_decode(response_tensors)
22     batch["ref_response"] = tokenizer.batch_decode(ref_response_tensors)
23
24     # Compute sentiment score
25     texts = [q + r for q, r in zip(batch["query"], batch["response"])]
26     pipe_outputs = sentiment_pipe(texts, **sent_kwargs)
27     rewards = [torch.tensor(output[1]["score"]) for output in pipe_outputs]
28     ref_texts = [q + r for q, r in zip(batch["query"], batch["ref_response"])]
29     ref_pipe_outputs = sentiment_pipe(ref_texts, **sent_kwargs)
30     ref_rewards = [torch.tensor(output[1]["score"]) for output in ref_pipe_outputs]
31     batch["ref_rewards"] = ref_rewards
32
33     # Run PPO step
34     stats = ppo_trainer.step(query_tensors, response_tensors, rewards)
35     ppo_trainer.log_stats(stats, batch, rewards, columns_to_log=["query", "response",
36     "ref_response", "ref_rewards"])
```

Phần 4. Câu hỏi trắc nghiệm

Câu hỏi 1 Phương pháp huấn luyện của mô hình ChatGPT là gì?

- a) Tinh chỉnh mô hình ngôn ngữ lớn GPT2
- b) Tinh chỉnh mô hình ngôn ngữ lớn BLOOM
- c) Tinh chỉnh mô hình mBART
- d) Tinh chỉnh mô hình ngôn ngữ lớn GPT 3.5(Large Language Model) với phương pháp học tăng cường dựa trên phản hồi người dùng.

Câu hỏi 2 Bộ dữ liệu để tinh chỉnh mô hình ChatGPT là gì?

- a) Dữ liệu tóm tắt văn bản
- b) Dữ liệu các cuộc hội thoại
- c) Dữ liệu phân loại văn bản
- d) Dữ liệu dịch máy

Câu hỏi 3 Phương pháp cài đặt nào không có trong phương pháp học trong ngữ cảnh (In-Content Learning) được sử dụng trong quá trình sinh văn bản là gì?

- a) One-shot
- b) Zero-shot
- c) One-shot
- d) Fine-Tuning

Câu hỏi 4 Thư viện nào sau đây hỗ trợ sử dụng ChatGPT API?

- a) Spacy
- b) Fasttext
- c) Langchain
- d) Fairseq

Câu hỏi 5 Dựa vào dữ liệu đầu vào, bài toán tóm tắt văn bản được chia thành những loại nào?

- a) Single-Document và Multi-Document
- b) Abstractive và Single-Document
- c) Extractive và Abstractive
- d) Multi-Document và Extractive

Câu hỏi 6 Dựa vào dữ liệu đầu ra, bài toán tóm tắt văn bản được chia thành những loại nào?

- a) Single-Document và Multi-Document
- b) Abstractive và Single-Document
- c) Extractive và Abstractive
- d) Multi-Document và Extractive

Câu hỏi 7 Độ đo đánh giá bài toán tóm tắt văn bản là?

- a) Accuracy
- b) F1
- c) BLEU
- d) ROUGE

Câu hỏi 8 Bộ dữ liệu sử dụng cho mô hình SFT là?

- a) CarperAI/openai_summarize_tldr
- b) MNIST
- c) CIFAR10
- d) CIFAR100

Câu hỏi 9 Bài toán nào sau đây sử dụng để huấn luyện mô hình SFT?

- a) Text Classification
- b) Machine Translation
- c) Text Generation
- d) Question Answering

Câu hỏi 10 Bài toán nào sau đây sử dụng để huấn luyện mô hình Reward?

- a) Text Classification
- b) Machine Translation
- c) Text Generation
- d) Question Answering

- *Hết* -