



Building Vietnamese Chatbot using LLMs and RLHF

LLMs Series: Project

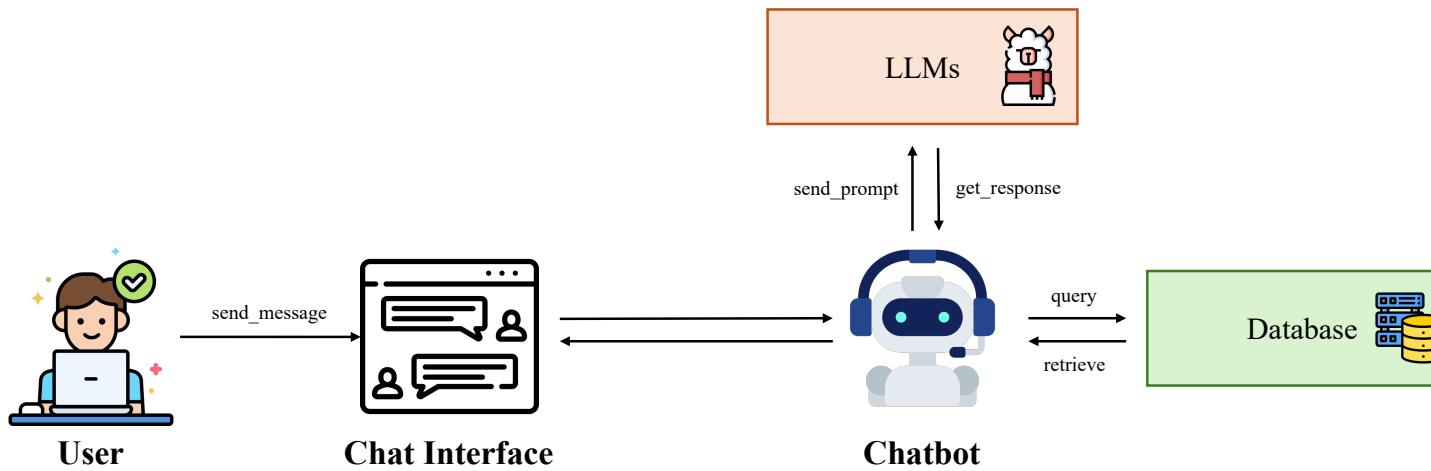
Nguyen-Thuan Duong – TA

Yen-Linh Vu – STA

Anh-Khoi Nguyen– STA

Objectives

❖ Content



In this study session, we will discuss about:

1. Introduction to Chatbot? What is Chatbot using LLMs?
2. How to fine-tuning LLMs for a Chatbot applications (conversation data).
3. How to improve response from a Chatbot (using LLMs) with RLHF?
4. How to implement a Chat Interface for a complete Chatbot application?

Outline

- Introduction
- Fine-tuning LLMs for Chatbot
- Training LLMs with RLHF
- Building Chat Interface
- Question

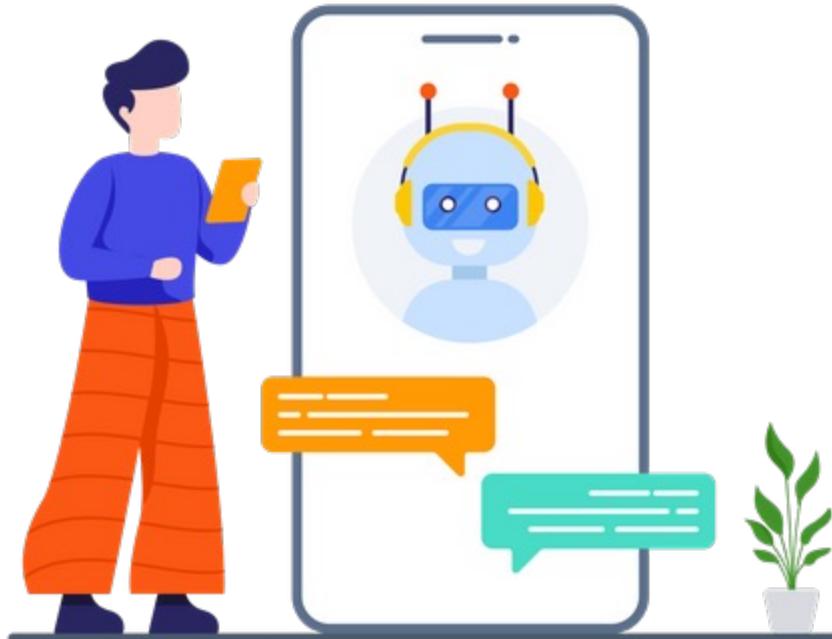
 AI

AI VIET NAM
@aivietnam.edu.vn

Introduction

Introduction

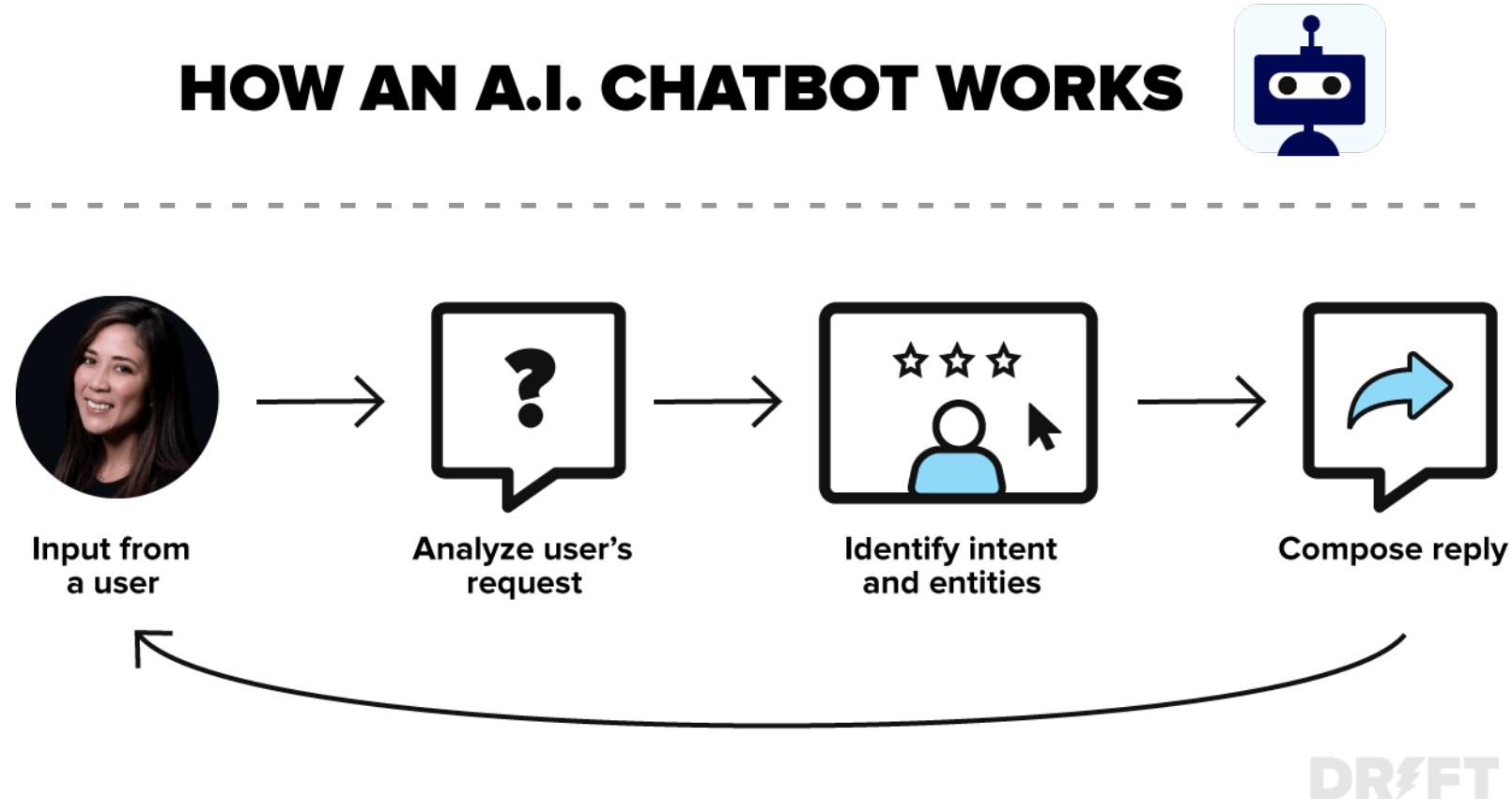
❖ What is a Chatbot?



Chatbot: A software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile apps, or through telephone.

Introduction

❖ How Chatbot works?



Introduction

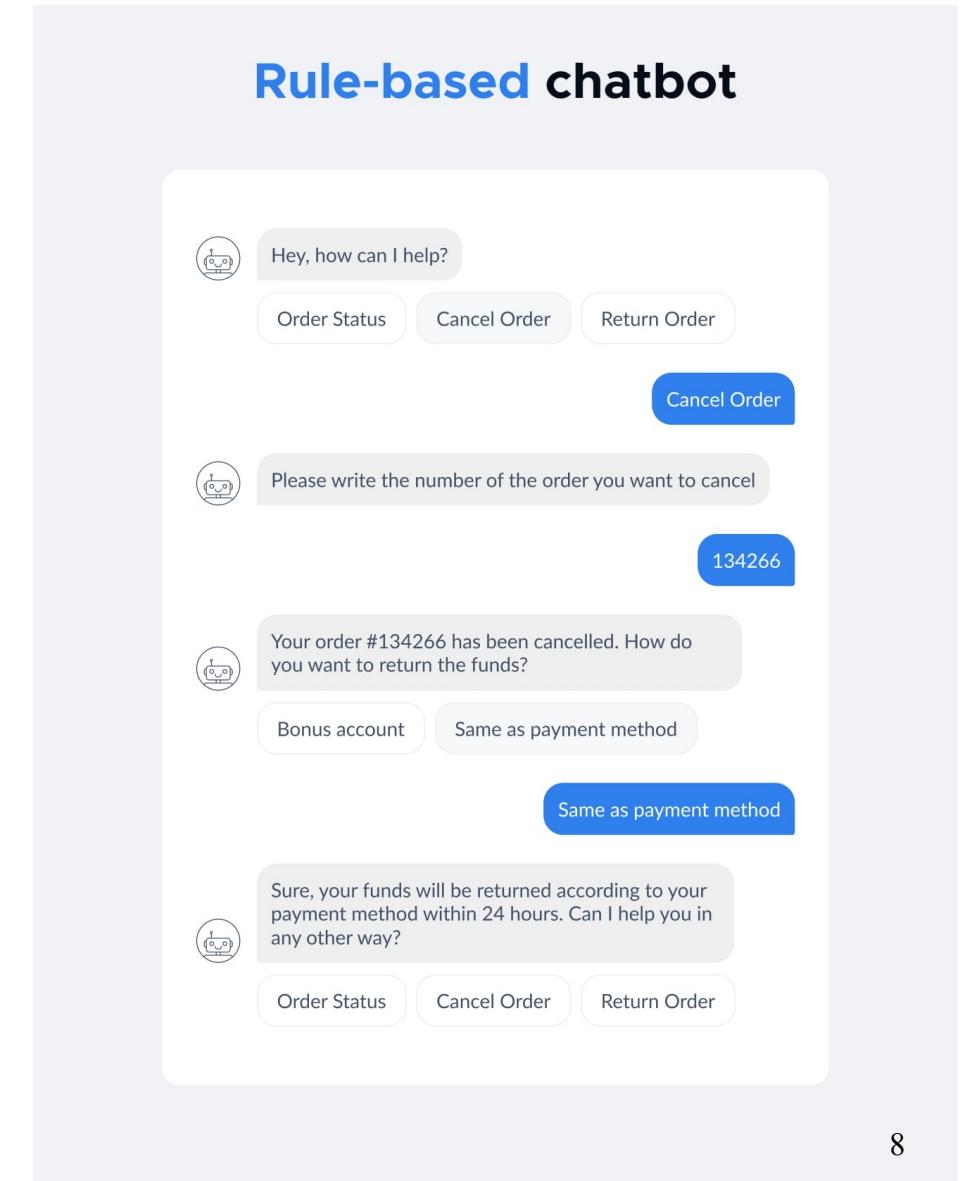
❖ Chatbot Categorization

Chatbot Categories	Knowledge Domain	Generic
		Open Domain
		Closed Domain
	Service Provided	Interpersonal
		Intrapersonal
		Inter-agent
	Goals	Informative
		Chat-based/Conversational
		Task-based
	Response Generation Method	Rule-based
		Retrieval-based
		Generative
	Human-aid	Human-mediated
		Autonomous
	Permissions	Open-source
		Commercial
	Communication Channel	Text
		Voice
		Image

Introduction

❖ Chatbot Categorization: Rule-based

Rule-based chatbots operate based on a predefined set of rules and patterns. They follow a strict set of if-then statements or decision trees to determine the appropriate response to a user's input.

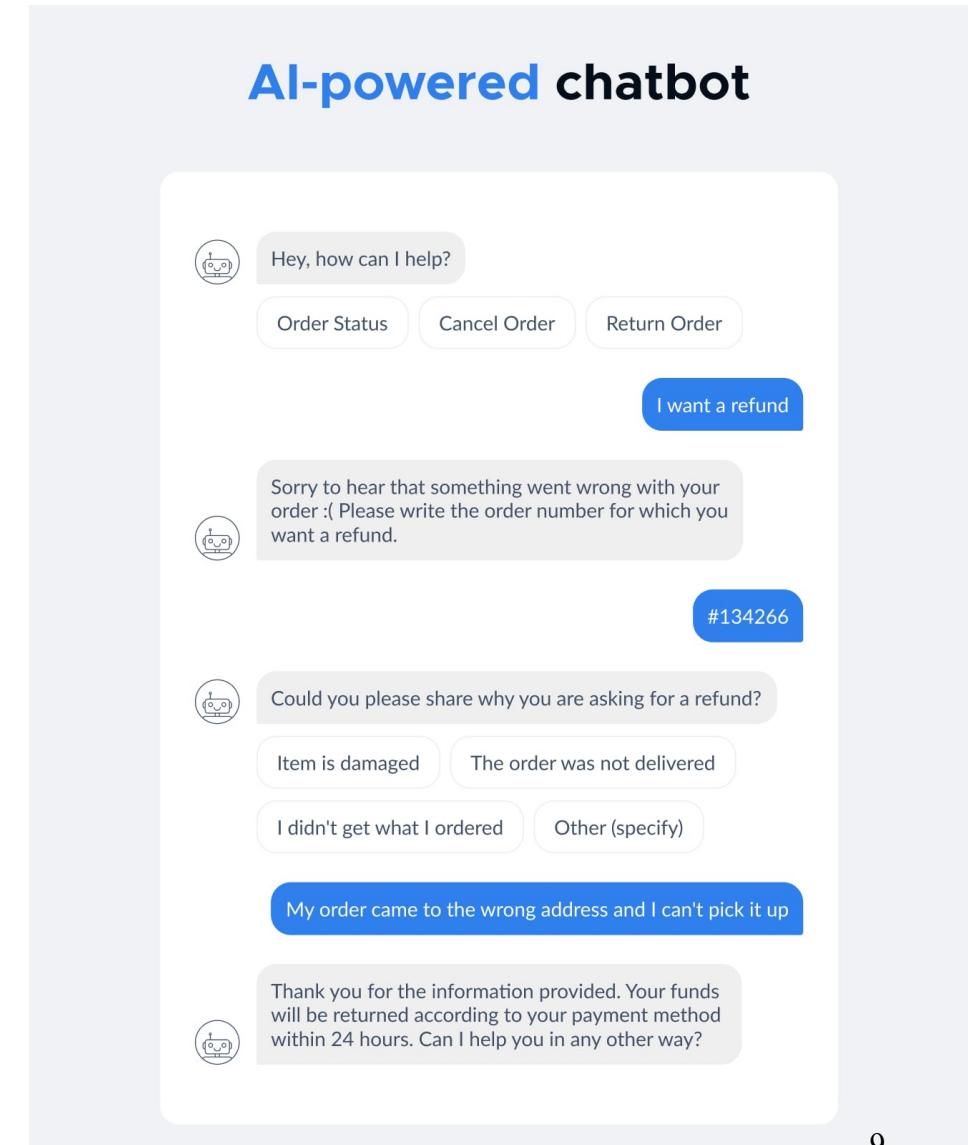




Introduction

❖ Chatbot Categorization: Retrieval-based

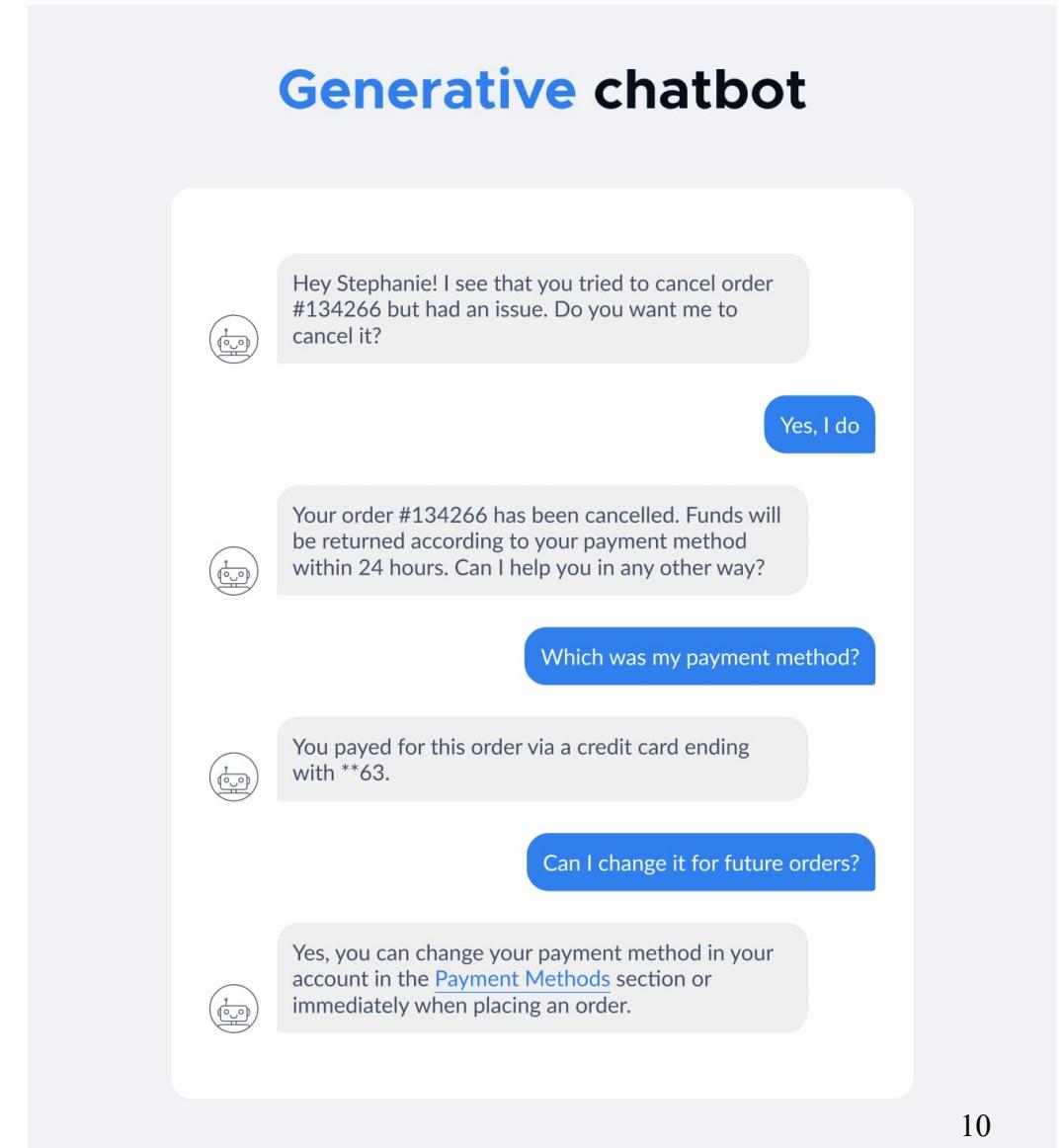
Retrieval-based chatbots are a type of artificial intelligence chatbot (AI-powered chatbot) that responds to user queries by selecting answers from a predefined set of responses. It uses algorithms to interpret and match the user's input to the most relevant answer in its database.



Introduction

❖ Chatbot Categorization: Generative

Generative chatbots, also known as language models or conversational AI models, are chatbots that generate responses to user inputs dynamically rather than relying on pre-defined responses or templates. Nowadays, this typically involves the use of LLMs.





Introduction

❖ Generative AI

ChatGPT 4o ▾

write a python function that print "Hello AI VIETNAM" 5 times

Here's a simple Python function that prints "Hello AI VIETNAM" five times:

```
python
def print_hello_ai_vietnam():
    for _ in range(5):
        print("Hello AI VIETNAM")
```

To use this function, just call it like this:

```
python
print_hello_ai_vietnam()
```

Would you like it to also return the messages as a list or save to a file?

Is this conversation helpful so far?

Ask anything

+ ...

ChatGPT can make mistakes. Check important info.



\$> gta jim halpert john
krasinski profile picture by
greg rutkowski, dynamic pose,
intricate, futuristic,
fantasy, elegant, by stanley
artgerm lau, greg rutkowski,
thomas kindkade, alphonse
muchá, loish, norman
rockwell, fantasy lut,
asymmetric, long hair, retro
computer graphics, video
game, fluid lines

\$> gta brad pitt john
krasinski profile picture by
greg rutkowski, dynamic pose,
intricate, futuristic,
fantasy, elegant, by stanley
artgerm lau, greg rutkowski,
thomas kindkade, alphonse
muchá, loish, norman
rockwell, fantasy lut,
asymmetric, long hair, retro
computer graphics, video
game, fluid lines

Introduction

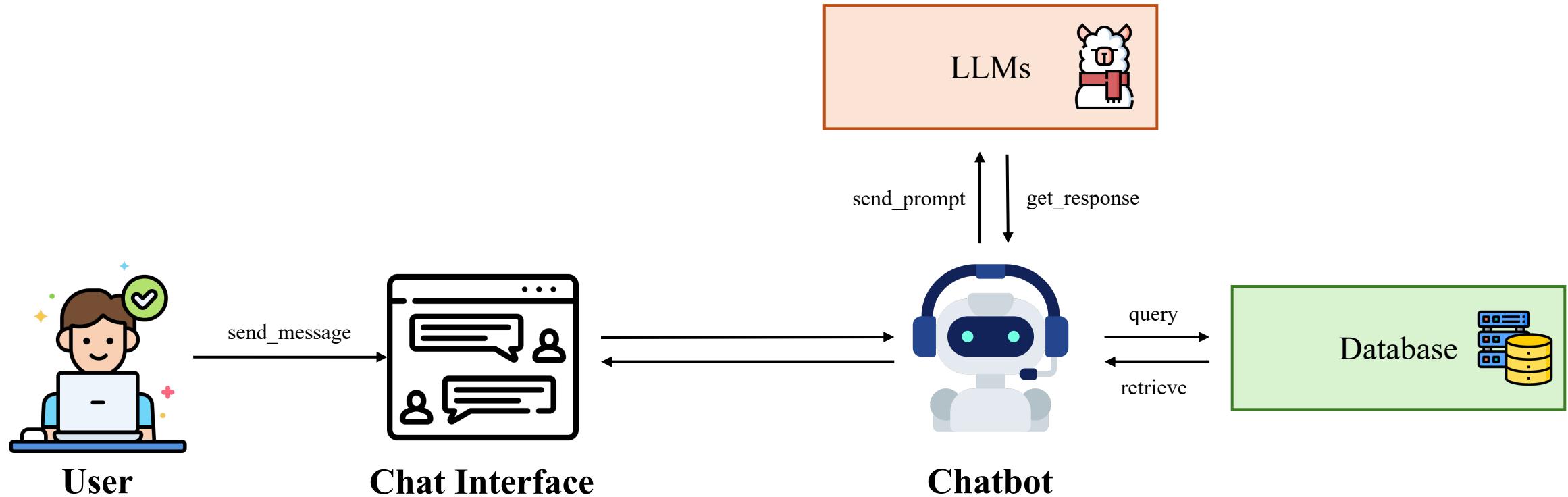
❖ Chatbot Categorization

Chatbot Categories	Knowledge Domain	Generic
		Open Domain
		Closed Domain
	Service Provided	Interpersonal
		Intrapersonal
		Inter-agent
	Goals	Informative
		Chat-based/Conversational
		Task-based
	Response Generation Method	Rule-based
		Retrieval-based
		Generative
	Human-aid	Human-mediated
		Autonomous
		Open-source
	Permissions	Commercial
		Text
		Voice
	Communication Channel	Image

We will implement a basic chatbot using LLMs.

Introduction

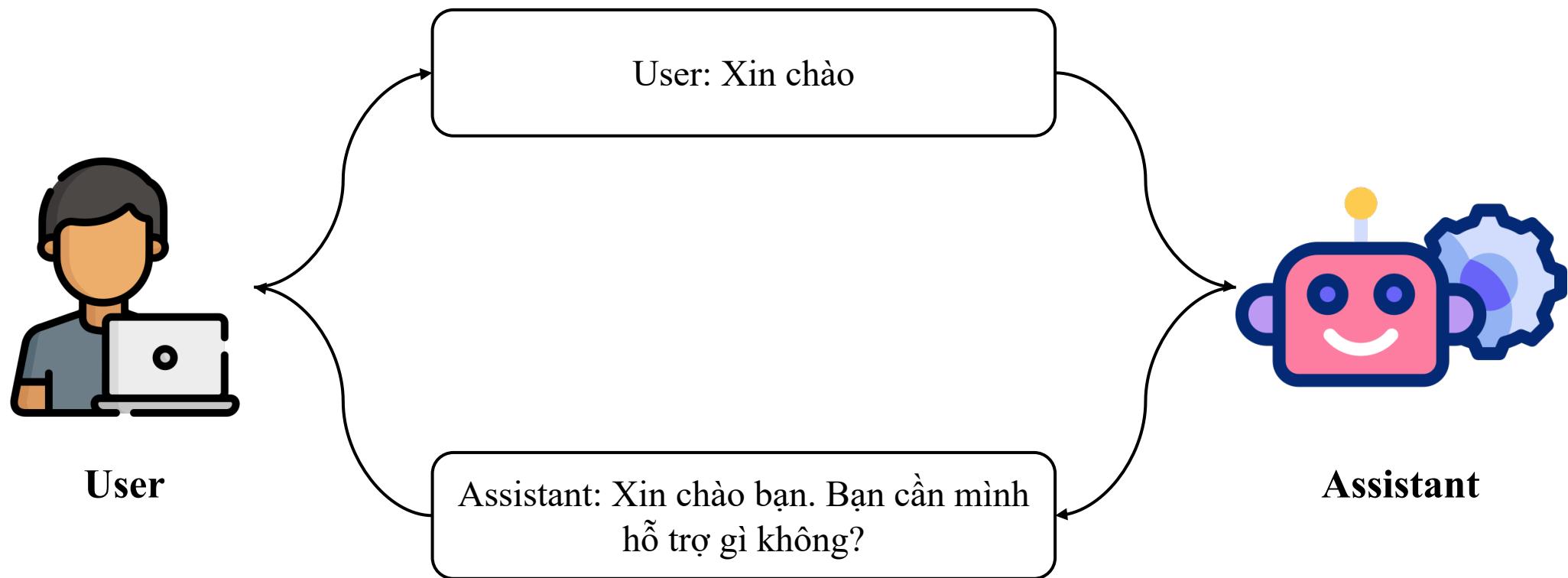
❖ Chatbot with LLMs



Introduction

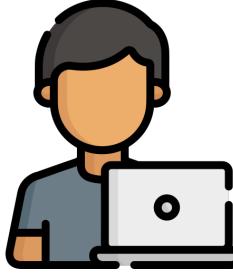
❖ Project Statement

Description: Build a Vietnamese Chatbot application by fine-tuning an LLM on Vietnamese conversation data and then further improve the response from the model by utilizing RLHF (Reinforcement Learning from Human Feedback).

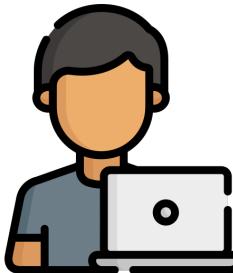


Introduction

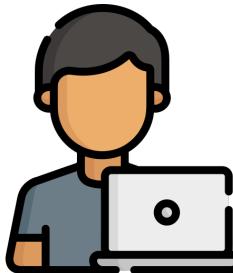
❖ Chatbot Response



Xin chào



Bạn giỏi về địa lý không?



Mình có kiến thức khá rộng về địa lý

Trường Sa, Hoàng Sa là của ai ?

Introduction

❖ Chatbot Response

Xin chào bạn, tôi là Thuận. Hãy dịch câu sau sang tiếng anh: "Tôi muốn mua một chiếc ô tô màu xanh lá"

Chào Thuận! Câu tiếng Anh của bạn là:
"I want to buy a green car."

Nếu bạn cần thêm cách diễn đạt khác hoặc muốn nói trang trọng hơn, cứ nói nhé!

"Tôi muốn mua một ly cà phê sữa"

Câu tiếng Anh là:
"I want to buy a cup of milk coffee."

Tuy nhiên, nếu bạn đang nói về cà phê sữa kiểu Việt Nam (cà phê pha với sữa đặc), cách nói tự nhiên hơn sẽ là:

"I want to buy a cup of Vietnamese milk coffee."

Input tokens

∞Llama 3.2

Output tokens

Introduction

❖ Language Model

Pre-training LLM

Next Token Prediction

Language Model

GPT-3, LLama-1B



Introduction

❖ SFT

Supervised Fine-Tuning
(SFT)

Instruction Following

Instruction Model

LLaMA-3.2-1B-Instruct



Introduction

❖ Fine-tuning with RL

RLHF

Preference Optimization

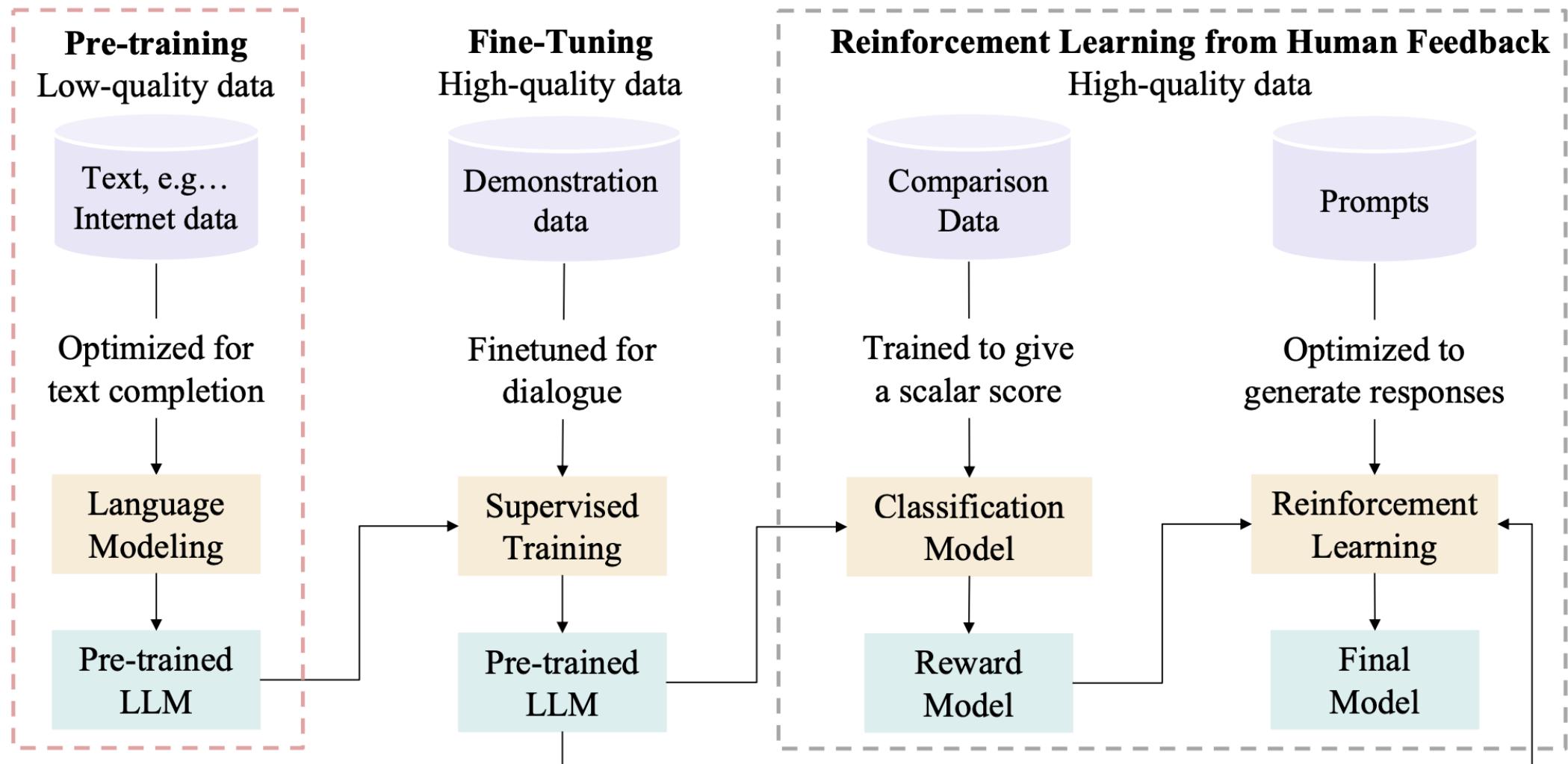
Aligned Model

ChatGPT



Introduction

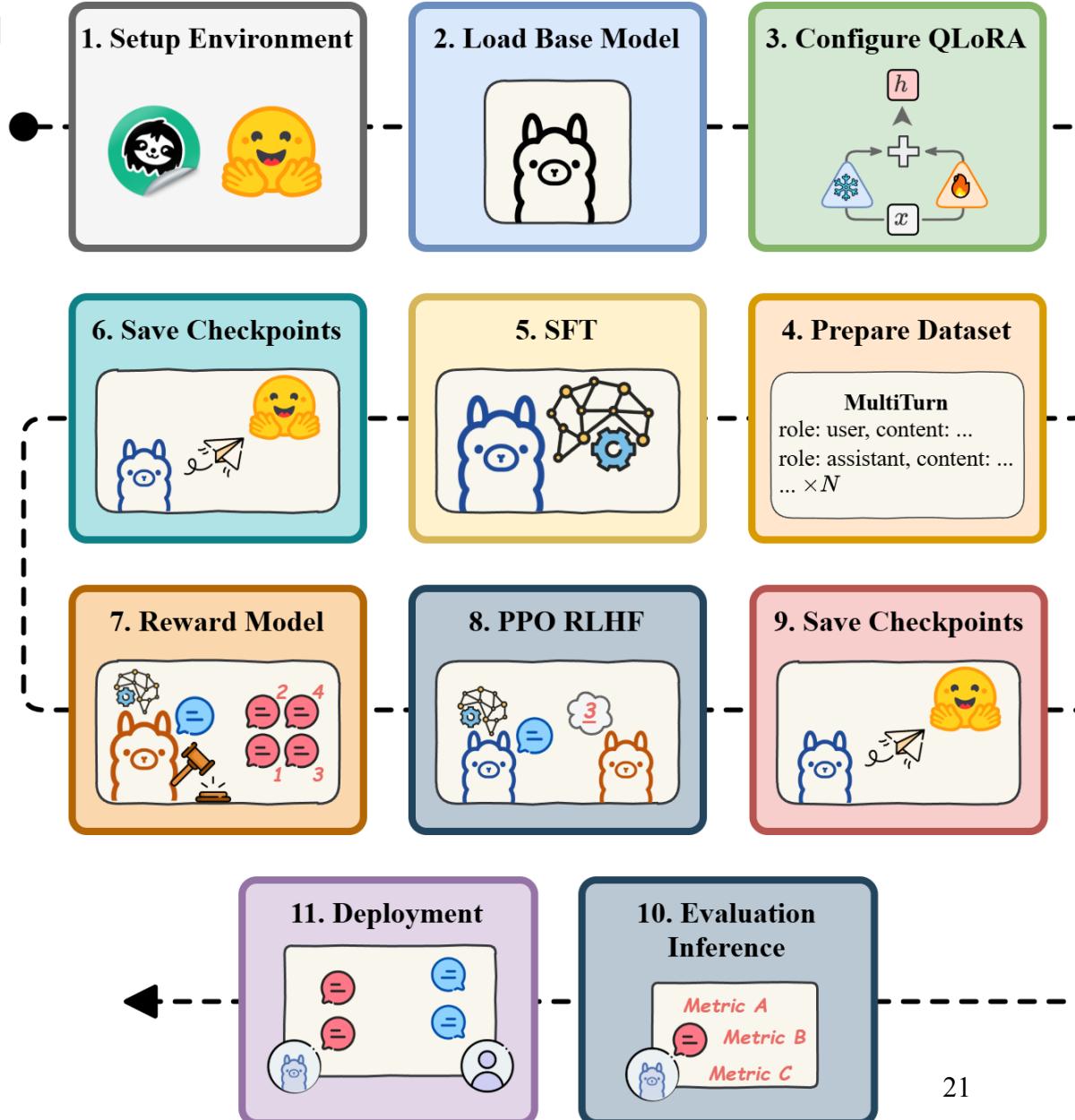
❖ RLHF



Introduction

❖ Project Pipeline

Description: Build a Vietnamese Chatbot application by fine-tuning an LLM on Vietnamese conversation data and then further improve the response from the model by utilizing RLHF (Reinforcement Learning from Human Feedback).



Fine-tuning LLMs for Chatbot

Fine-tuning LLMs for Chatbot

❖ Setup Environment - Unsloth

1. Setup Environment



<https://unsloth.ai/>

<https://github.com/unslotha/unsloth>

```
1 !pip install -q unsloth==2025.4.7 datasets==3.5.1
```

```
1 import torch
2 from trl import SFTTrainer
3 from datasets import load_dataset
4 from unsloth import FastLanguageModel
5 from unsloth import FastLanguageModel
6 from transformers import GenerationConfig
7 from transformers import TrainingArguments
```

We tested using the Alpaca Dataset, a batch size of 2, gradient accumulation steps of 4, rank = 32, and applied QLoRA on all linear layers (q, k, v, o, gate, up, down):

Model	VRAM	👉 Unsloth speed	👉 VRAM reduction	👉 Longer context	😊 Hugging Face + FA2
Llama 3.3 (70B)	80GB	2x	>75%	13x longer	1x
Llama 3.1 (8B)	80GB	2x	>70%	12x longer	1x

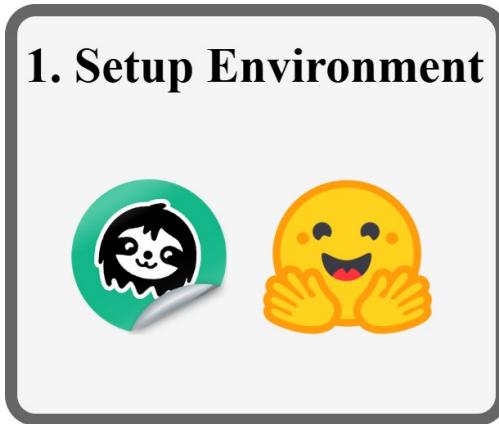
We tested Llama 3.3 (70B) Instruct on a 80GB A100 and did 4bit QLoRA on all linear layers (Q, K, V, O, gate, up and down) with rank = 32 with a batch size of 1. We padded all sequences to a certain maximum sequence length to mimic long context finetuning workloads.

GPU VRAM	👉 Unsloth context length	Hugging Face + FA2
48 GB	12,106	OOM
80 GB	89,389	6,916



Fine-tuning LLMs for Chatbot

❖ Setup Environment - Login



Access Tokens

User Access Tokens

Create new Access Token

Token type: Fine-grained (Read, Write)

This cannot be changed after token creation.

Token name: Create token

Name	Value	Last Refreshed Date	Last Used Date	Permissions
My token	hf_...nSKL	1 minute ago	-	FINEGRAINED

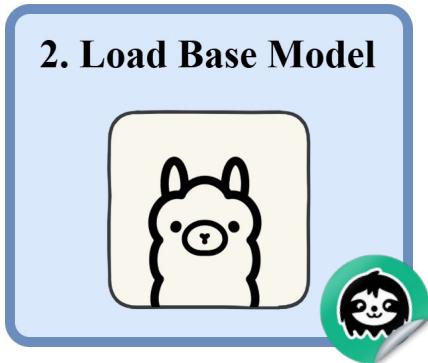
Hugging Face

<https://huggingface.co/settings/tokens>

```
1 login(token="hf_xxxxxxxxxxxxxxxxxxxxxx")
```

Fine-tuning LLMs for Chatbot

❖ Initialize Model



Unsloth supports	Free Notebooks	Performance	Memory use
Qwen3 (14B)	Start for free	2x faster	70% less
GRPO (R1 reasoning)	Start for free	2x faster	80% less
Gemma 3 (4B)	Start for free	1.6x faster	60% less
Llama 3.2 (3B)	Start for free	2x faster	70% less
Phi-4 (14B)	Start for free	2x faster	70% less
Llama 3.2 Vision (11B)	Start for free	2x faster	50% less
Llama 3.1 (8B)	Start for free	2x faster	70% less
Mistral v0.3 (7B)	Start for free	2.2x faster	75% less
Ollama	Start for free	1.9x faster	60% less
DPO Zephyr	Start for free	1.9x faster	50% less

```
1 MAX_SEQ_LENGTH = 2048
2
3 model, tokenizer = FastLanguageModel.from_pretrained(
4     model_name="unsloth/Llama-3.2-1B-Instruct-bnb-4bit",
5     max_seq_length=MAX_SEQ_LENGTH,
6     load_in_4bit=True,
7     dtype=torch.bfloat16,
8 )
```

Fine-tuning LLMs for Chatbot

❖ Apply QLoRA

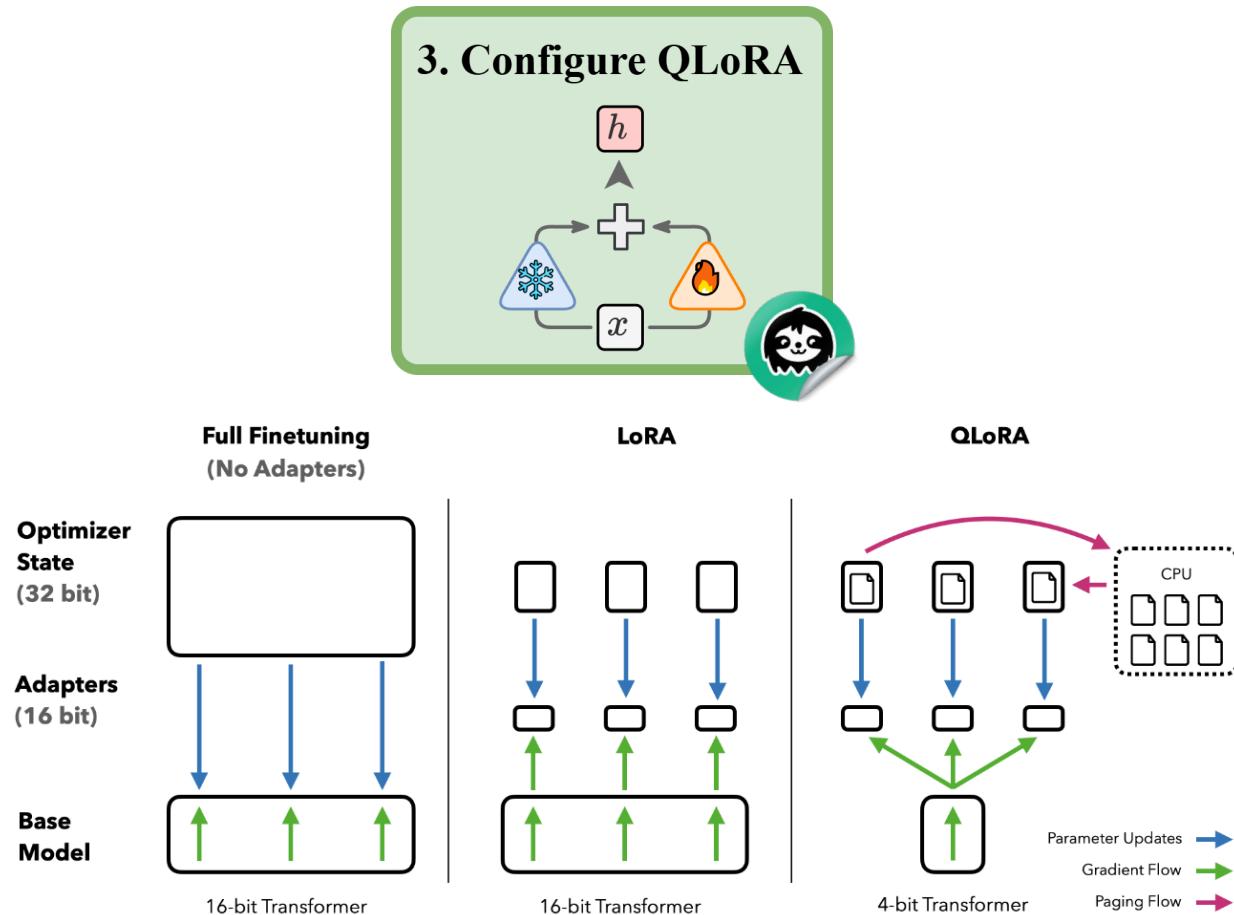


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

```

10 model = FastLanguageModel.get_peft_model(
11     model,
12     r=16,
13     lora_alpha=16,
14     lora_dropout=0,
15     target_modules=[
16         "q_proj", "k_proj", "v_proj", "up_proj",
17         "down_proj", "o_proj", "gate_proj"],
18     use_rslora=True,
19     use_gradient_checkpointing="unslosh",
20     random_state = 42,
21 )

```

```
1 model.print_trainable_parameters()
```

```
trainable params: 11,272,192 || all params: 1,247,086,592
```

```
trainable%: 0.9039
```

Fine-tuning LLMs for Chatbot

❖ Prepare Dataset - Introduce

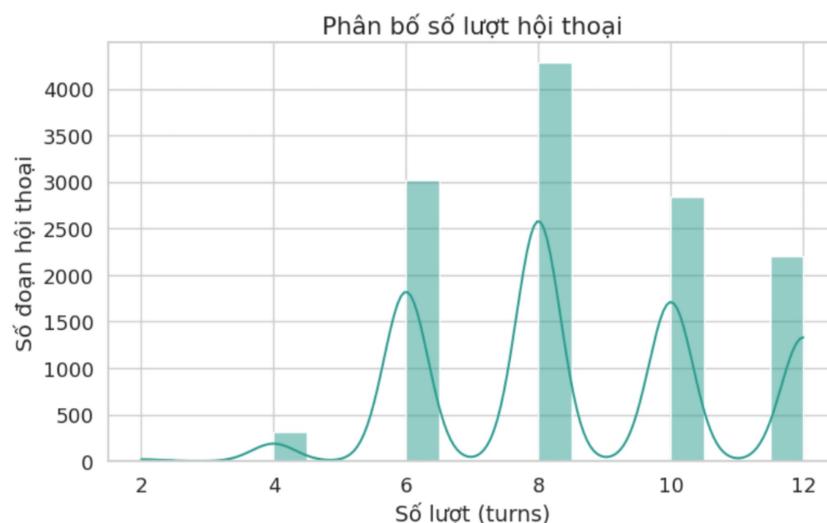
4. Prepare Dataset

MultiTurn

role: user, content: ...

role: assistant, content: ...

... $\times N$



<https://huggingface.co/datasets/5CD-AI/Vietnamese-Multi-turn-Chat-Alpaca>

Split (1)

train · 12.7k rows

Search this dataset

conversations

list · lengths



[{ "from": "human", "value": "Hãy chỉnh sửa câu này để ngắn gọn hơn mà không mất đi ý nghĩa: \"Trận đấu là một thất bại nặng nề mặc dù thực tế là cả đội đã tập luyện trong nhiều tuần.\""}]

[{ "from": "human", "value": "Tạo một câu hỏi bằng tiếng Anh liên quan đến chủ đề đã cho: Biển đổi khí hậu" }, { "from": "gpt", "value": "Nguyên nhân của biến đổi khí hậu là gì?" }, { }]

[{ "from": "human", "value": "Phân loại sản phẩm là \"dụng cụ\" hay \"đồ chơi\": Ô tô điều khiển từ xa" }, { "from": "gpt", "value": "Dụng cụ" }, { "from": "human", "value": "Bạn có thể cho..." }]

[{ "from": "human", "value": "Với đoạn này, hãy tóm tắt các điểm chính của nó: Màn hình máy tính hiện đại có lợi thế hơn màn hình CRT truyền thống, chẳng hạn như kích thước nhỏ hơn, mức...

[{ "from": "human", "value": "Câu sau đây đúng hay sai? Con người có thể thở dưới nước." }, { "from": "gpt", "value": "Sai - con người không thể thở dưới nước." }, { "from": "human", "value": "..."}]

[{ "from": "human", "value": "Tạo ra một câu chuyện dành cho trẻ em trong đó có nhân vật một chú sư tử thân thiện." }, { "from": "gpt", "value": "Ngày xưa ngày xưa, sâu trong rừng rậm có..." }]

< Previous 1 2 3 ... 127 Next >

```
1 dataset_name = "5CD-AI/Vietnamese-Multi-turn-Chat-Alpaca"
2 raw_dataset = load_dataset(dataset_name, split="train")
```

Fine-tuning LLMs for Chatbot

❖ Prepare Dataset - Process

```
3 SYS_INSTRUCT = "Bạn là một trợ lý AI thân thiện, hãy trả lời bằng tiếng Việt."
4
5 def convert_to_chat_format(conversations):
6     messages = [{"role": "system", "content": SYS_INSTRUCT}]
7     for msg in conversations:
8         role = "user" if msg["from"] == "human" else "assistant"
9         messages.append({"role": role, "content": msg["value"]})
10    return messages
```

[{"from": "human", "value": "Câu sau đây đúng hay sai? Con người có thể thở dưới nước."}, {"from": "gpt", "value": "Sai - con người không thể thở dưới nước."}, {"from": "human", "value": "Động vật nào có thể thở dưới nước mà không cần bình oxy?"}, {"from": "gpt", "value": "Có nhiều loại động vật có khả năng thở dưới nước mà không cần bình oxy. Những động vật này được gọi là động vật thủy sinh hay \"động vật biển\" và bao gồm cá, cá voi, cá heo, hải cẩu, sư tử biển, rùa, cá sấu, cá sấu Mỹ và nhiều loài động vật không xương sống như bạch tuộc và trai. Những loài động vật này đã thích nghi với việc sống và thở dưới nước qua hàng triệu năm tiến hóa."}, {"from": "human", "value": "Những loài động vật nào sau đây có khả năng thở dưới nước bằng da?"}, {"from": "gpt", "value": "Có một số loài động vật có thể thở qua da, quá trình này được gọi là hô hấp qua da. Ví dụ về động vật có khả năng hô hấp ở da bao gồm động vật lưỡng cư như ếch và kỳ nhông, một số loại cá và một số loài rắn biển. Hô hấp qua da cũng được sử dụng bởi một số côn trùng, động vật giáp xác và động vật không xương sống khác. Tuy nhiên, chưa có trường hợp nào được biết đến về việc động vật có vú có thể thở bằng da."}, {"from": "human", "value": "Bạn có thể cho một số ví dụ về loài cá có thể thở bằng da không?"}, {"from": "gpt", "value": "Đúng, có một số ví dụ về các loài cá có thể thở qua da, bao gồm: 1. Lươn điện - Lươn điện có miệng có nhiều mạch máu và sử dụng da để hấp thụ oxy từ nước. 2. Cá da trơn - Một số loài cá da trơn, chẳng hạn như cá da tron biết đi, có khả năng thở qua da, nơi có nhiều mạch máu và cho phép trao đổi khí. 3. Cá phổi - Cá phổi có cả mang và phổi và có thể thở bằng da khi chìm trong nước cũng như khi tiếp xúc với không khí. 4. Cá betta - Cá betta có cơ quan mề cung, một cấu trúc hô hấp chuyên biệt cho phép chúng lấy oxy từ không khí và nước qua da. 5. Cá Koi - Cá Koi có mang và da có nhiều mạch máu cho phép chúng lấy oxy từ nước. Những con cá này đã thích nghi để lấy oxy từ môi trường xung quanh theo những cách khác nhau, với làn da của chúng hoạt động như một cơ quan hô hấp quan trọng ngoài mang hoặc phổi."}, {...}, {...}]



Fine-tuning LLMs for Chatbot

❖ Prepare Dataset - Process

```
12 def format_prompt(example):
13     messages = convert_to_chat_format(example["conversations"])
14     return {
15         "text": tokenizer.apply_chat_template(
16             messages,
17             tokenize=False,
18             add_generation_prompt=False
19         )
20     }
```

```
1 chat_template = tokenizer.apply_chat_template(
2 [
3     {"role": "system", "content": "Bạn là một trợ lý AI thân thiện, hãy trả lời bằng tiếng Việt."},
4     {"role": "user", "content": "Câu sau đây đúng hay sai? Con người có thể thở dưới nước."},
5     {"role": "assistant", "content": "Sai - con người không thể thở dưới nước."},
6     # ...
7 ],
8 tokenize=False,
9 add_generation_prompt=False,
10 )
11
12
13 print(chat_template)
```

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date: December 2023
Today Date: 06 May 2025

Bạn là một trợ lý AI thân thiện, hãy trả lời bằng tiếng Việt.<|eot_id|><|start_header_id|>user<|end_header_id|>
Câu sau đây đúng hay sai? Con người có thể thở dưới nước.<|eot_id|><|start_header_id|>assistant<|end_header_id|>
Sai - con người không thể thở dưới nước.<|eot_id|>

```
[{"from": "human", "value": "Câu sau đây đúng hay sai? Con\nngười có thể thở dưới nước." },
{"from": "gpt", "value": "Sai - con người không thể thở dưới\nnước." },
{"from": "human", "value": "Động vật nào có thể thở dưới nước\nmà không cần bình oxy?" },
{"from": "gpt", "value": "Có nhiều loài động vật có khả năng thở\ndưới nước mà không cần bình oxy. Những động vật này được gọi là\nđộng vật thủy sinh hay \"động vật biển\" và bao gồm cá, cá voi, cá\nheo, hải cẩu, sư tử biển, rùa, cá sấu, cá sấu Mỹ và nhiều loài động\nvật không xương sống như bạch tuộc và trai. Những loài động vật\nnày đã thích nghi với việc sống và thở dưới nước qua hàng triệu\nnăm tiến hóa." },
{...},
{...}]
```

Fine-tuning LLMs for Chatbot

❖ Prepare Dataset - Process

```
22 def tokenize_function(examples):
23     return tokenizer(
24         examples["text"],
25         truncation=True,
26         max_length=MAX_SEQ_LENGTH,
27         padding="max_length",
28     )
29
30 dataset = raw_dataset.map(format_prompt, remove_columns=raw_dataset.column_names)
31 dataset = dataset.map(tokenize_function, batched=True)
```

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
Cutting Knowledge Date: December 2023
Today Date: 06 May 2025

Bạn là một trợ lý AI thân thiện, hãy trả lời bằng tiếng Việt.<|eot_id|><|start_header_id|>user<|end_header_id|>

Câu sau đây đúng hay sai? Con người có thể thở dưới nước.<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Sai - con người không thể thở dưới nước.<|eot_id|>

Fine-tuning LLMs for Chatbot

❖ Supervised Fine-tuning



```
--((=====))-- Unsloth - 2x faster free finetuning | Num GPUs used = 1
    \ \ /| Num examples = 12,697 | Num Epochs = 5 | Total steps = 400
0^0/ \_/\ \ Batch size per device = 64 | Gradient accumulation steps = 2
\       / Data Parallel GPUs = 1 | Total batch size (64 x 2 x 1) = 128
"-____-" Trainable parameters = 11,272,192/1,000,000,000 (1.13% trained)
Unsloth: Will smartly offload gradients to save VRAM!
[400/400 3:34:01, Epoch 4/5]
```

Step Training Loss

100	1.527100
200	1.349700
300	1.299000
400	1.277400

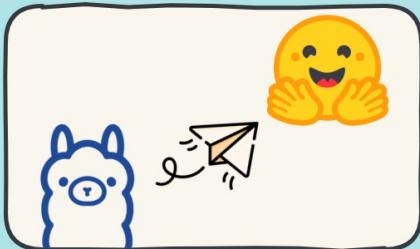
```
1 training_args = TrainingArguments(
2     per_device_train_batch_size=32,
3     gradient_accumulation_steps=2,
4     learning_rate=1e-4,
5     save_total_limit=4,
6     logging_steps=20,
7     output_dir=". ./checkpoint/llama3-1b-multi-conversation",
8     optim="paged_adamw_8bit",
9     lr_scheduler_type="cosine",
10    warmup_ratio=0.05,
11    save_strategy="steps",
12    save_steps=50,
13    report_to="none",
14    remove_unused_columns=True,
15    max_steps=400,
16    bf16=True,
17 )
```

```
19 trainer=SFTTrainer(
20     model=model,
21     args=training_args,
22     train_dataset=dataset,
23     tokenizer=tokenizer,
24 )
25
26 trainer.train()
```

Fine-tuning LLMs for Chatbot

❖ Save Model

6. Save Checkpoints



```
1 model, tokenizer = FastLanguageModel.from_pretrained(  
2     "./checkpoint/llama3-1b-multi-conversation/checkpoint-400",  
3     max_seq_length=2048,  
4     load_in_4bit=True,  
5     dtype=torch.bfloat16,  
6 )
```

```
1 model.push_to_hub_merged(  
2     "thuanan/Llama-3.2-1B-Instruct-Chat-sft",  
3     commit_message="Merge weights to push to hub",  
4 )
```

```
1 tokenizer.push_to_hub(  
2     "thuanan/Llama-3.2-1B-Instruct-Chat-sft",  
3     commit_message="Push tokenizer to hub",  
4 )
```



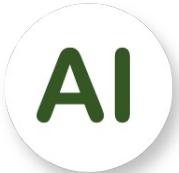
Fine-tuning LLMs for Chatbot

❖ Inference

```
12 prompt = [
13     {"role": "system", "content": (
14         "Bạn là một trợ lý AI thân thiện, "
15         "hãy trả lời bằng tiếng Việt."
16     ), 
17
18     {"role": "user", "content": (
19         "Hãy chỉnh sửa câu này để ngắn gọn hơn mà không mất đi ý nghĩa: "
20         "\"Trận đấu là một thất bại nặng nề "
21         "mặc dù thực tế là cả đội đã tập luyện trong nhiều tuần.\""
22     ), 
23
24     {"role": "assistant", "content": (
25         "Nhiều tuần huấn luyện của đội đã dẫn đến một thất bại nặng nề."
26     ), 
27
28     {"role": "user", "content": (
29         "Bạn có thể đề xuất một số chiến lược mà "
30         "nhóm có thể sử dụng để cải thiện hiệu suất "
31         "của họ trong trận đấu tiếp theo không?"
32     )}, 
33 ]
```

Assistant reply:

Chắc chắn! Dưới đây là một vài mẹo về cách các nhà tổ chức và cầu thủ team có thể áp dụng cho trò chơi sắp tới: - Đảm bảo rằng tất cả những người tham gia đều được đào tạo đầy đủ trước khi bắt đầu cuộc đua hoặc giải trí mới nhất. Điều quan trọng cần lưu tâm là việc chuẩn bị kỹ lưỡng sẽ giúp giảm thiểu rủi ro xảy ra sự cố trên sân khấu cũng như đảm bảo mọi thứ diễn ra suôn sẻ với tư duy sáng suốt tốt. - Tìm kiếm phản hồi từ đồng nghiệp và giám sát từng bước hành động của bạn nhằm xác định

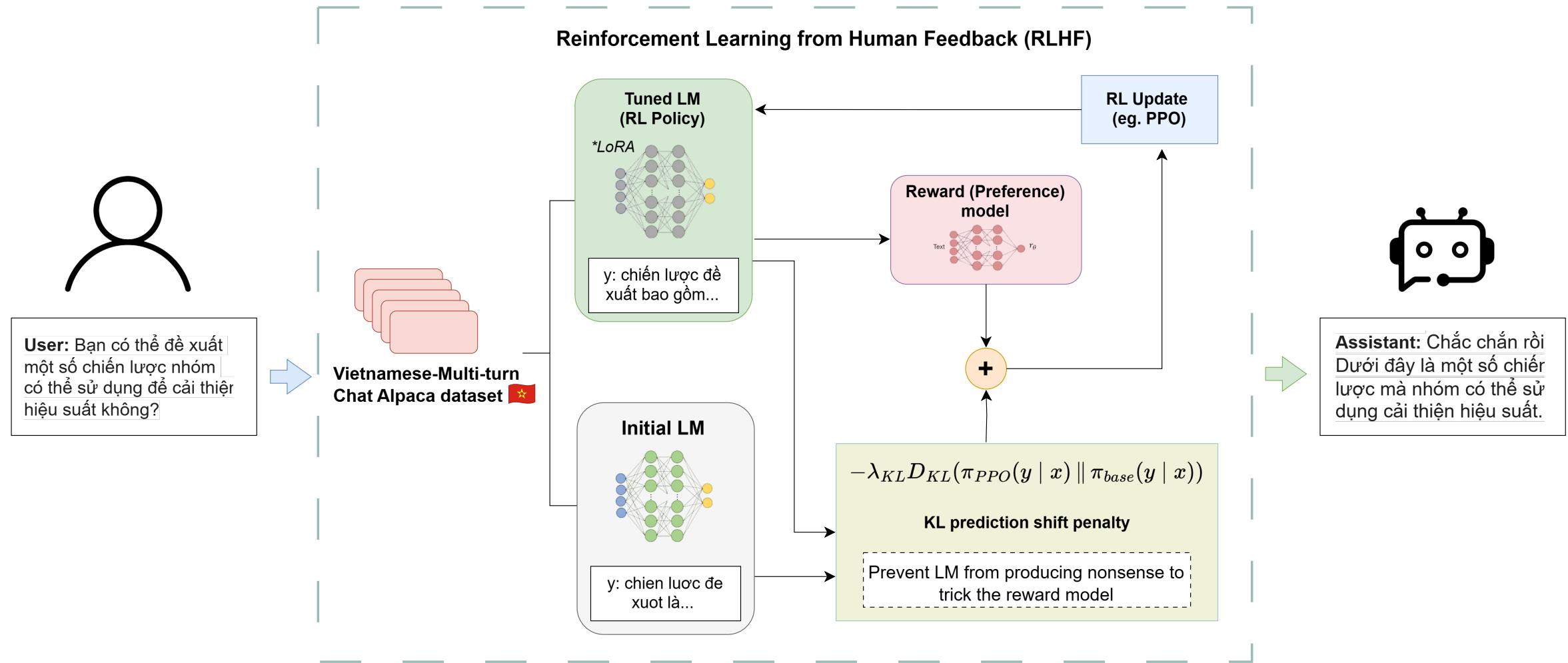
 AI

AI VIET NAM
@aivietnam.edu.vn

Training LLMs with RLHF

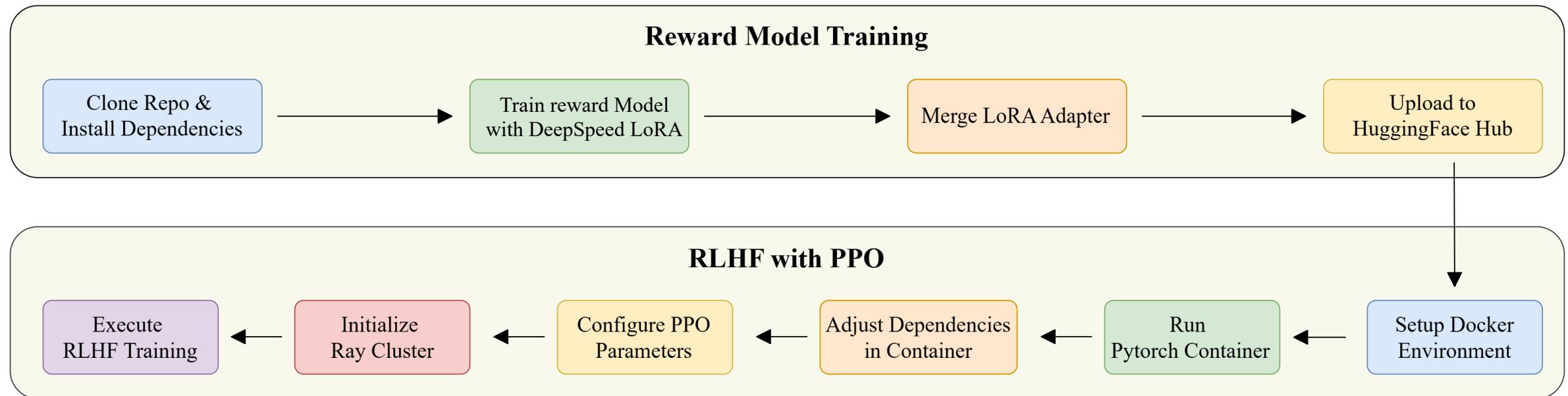
Training LLMs with RLHF

❖ Getting Started



Training LLMs with RLHF

❖ Pipeline





Training LLMs with RLHF

❖ Feedback

How to push a trained model to Huggingface Hub



You're giving feedback on a new version of ChatGPT.
Which response do you prefer? Responses may take a moment to load.

Response 1

To push a dataset to the Hugging Face Hub, follow these general steps using the `datasets` and `huggingface_hub` libraries:

✓ Step-by-step: Push to Hub (Datasets)

1. Install Required Libraries

```
bash                                ⚒ Copy ⚒ Edit
pip install datasets huggingface_hub
```

2. Login to Hugging Face CLI

```
bash                                ⚒ Copy ⚒ Edit
huggingface-cli login
```

Response 2

To push a dataset to the Hugging Face Hub, follow these steps. This assumes you're using Python and the 🤗 `datasets` library.

✓ 1. Install Required Libraries

```
bash                                ⚒ Copy ⚒ Edit
pip install datasets huggingface_hub
```

✓ 2. Login to Hugging Face CLI

```
bash                                ⚒ Copy ⚒ Edit
huggingface-cli login
```

Ask anything

Which response do you prefer?
Your choice will help make ChatGPT better.



Training LLMs with RLHF

❖ Collect feedback

You're giving feedback on a new version of ChatGPT.
Which response do you prefer? Responses may take a moment to load.

Response 1

To push a dataset to the Hugging Face Hub, follow these general steps using the `datasets` and `huggingface_hub` libraries:

✓ Step-by-step: Push to Hub (Datasets)

1. Install Required Libraries

```
bash
pip install datasets huggingface_hub
```

2. Login to Hugging Face CLI

```
bash
huggingface-cli login
```

Response 2

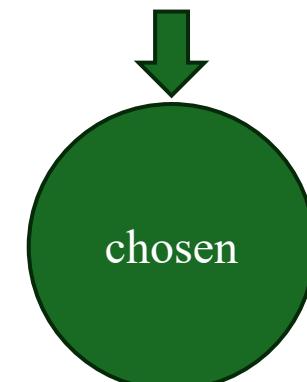
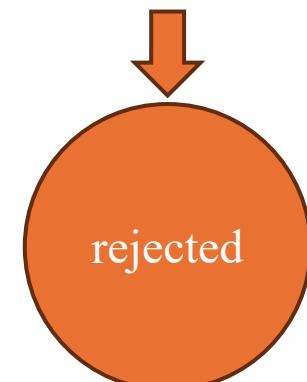
To push a dataset to the Hugging Face Hub, follow these steps. This assumes you're using Python and the 🤗 `datasets` library.

✓ 1. Install Required Libraries

```
bash
pip install datasets huggingface_hub
```

✓ 2. Login to Hugging Face CLI

```
bash
```

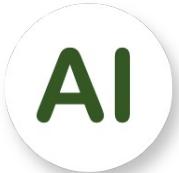




Training LLMs with RLHF

❖ Collect feedback

id string · lengths	chosen list · lengths	rejected list · lengths	prompt string · lengths	chosen_score null	rejected_score null
11..12	83.6%	2	100%		
alpaca-7294	[{ "content": "Xác định và sửa lỗi ngữ pháp.\n\nTôi đã di đến cửa hàng.", "role": "user" }, { "content": "Không có lỗi ngữ pháp. Câu này đã chính xác.", "role": "assistant" }]	[{ "content": "Xác định và sửa lỗi ngữ pháp.\n\nTôi đã di đến cửa hàng.", "role": "user" }, { "content": "Câu này không có lỗi ngữ pháp.", "role": "assistant" }]	Xác định và sửa lỗi ngữ pháp. Tôi đã di đến cửa hàng.	null	null
alpaca-33460	[{ "content": "Tạo một dòng thời gian chi tiết cho một dự án phát triển phần mềm.\n\nTổng quan dự án: Phát triển ứng dụng di động..." }]	[{ "content": "Tạo một dòng thời gian chi tiết cho một dự án phát triển phần mềm.\n\nTổng quan dự án: Phát triển ứng dụng di động..." }]	Tạo một dòng thời gian chi tiết cho một dự án phát triển phần mềm. Tổng quan dự án: Phát triển ứng dụng di động cho hệ thống thanh...	null	null
alpaca-4214	[{ "content": "Một điều mà chúng ta luôn phải xin phép trước khi chia sẻ nội dung của người khác trực tuyến là gì?", "role": "user" }]	[{ "content": "Một điều mà chúng ta luôn phải xin phép trước khi chia sẻ nội dung của người khác trực tuyến là gì?", "role": "user" }]	Một điều mà chúng ta luôn phải xin phép trước khi chia sẻ nội dung của người khác trực tuyến là gì?	null	null
alpaca-29036	[{ "content": "Tranh luận ủng hộ chính sách bao gồm tất cả trong khuôn viên trường đại học.", "role": "user" }, { "content": "Đầu..." }]	[{ "content": "Tranh luận ủng hộ chính sách bao gồm tất cả trong khuôn viên trường đại học.", "role": "user" }, { "content": "1..." }]	Tranh luận ủng hộ chính sách bao gồm tất cả trong khuôn viên trường đại học.	null	null
dolly-3046	[{ "content": "Các thành phần chính và gia vị đi vào nấu ăn Hy Lạp là gì?", "role": "user" }, { "content": "Các thành phần chính..." }]	[{ "content": "Các thành phần chính và gia vị đi vào nấu ăn Hy Lạp là gì?", "role": "user" }, { "content": "1. Rau củ tươi: cà..." }]	Các thành phần chính và gia vị đi vào nấu ăn Hy Lạp là gì?	null	null
alpaca-10039	[{ "content": "Ké tên 3 kỹ năng cần thiết để trở thành một kỹ sư phần mềm thành công", "role": "user" }, { "content": "1. Kiến thức..." }]	[{ "content": "Ké tên 3 kỹ năng cần thiết để trở thành một kỹ sư phần mềm thành công", "role": "user" }, { "content": "1. Kỹ năng..." }]	Ké tên 3 kỹ năng cần thiết để trở thành một kỹ sư phần mềm thành công	null	null
alpaca-25480	[{ "content": "Hãy tưởng tượng một nhân vật trái ngược với một siêu anh hùng điển hình và tạo ra một cốt truyện cho nhân vật..." }]	[{ "content": "Hãy tưởng tượng một nhân vật trái ngược với một siêu anh hùng điển hình và tạo ra một cốt truyện cho nhân vật..." }]	Hãy tưởng tượng một nhân vật trái ngược với một siêu anh hùng điển hình và tạo ra một cốt truyện cho nhân vật này.	null	null



AI

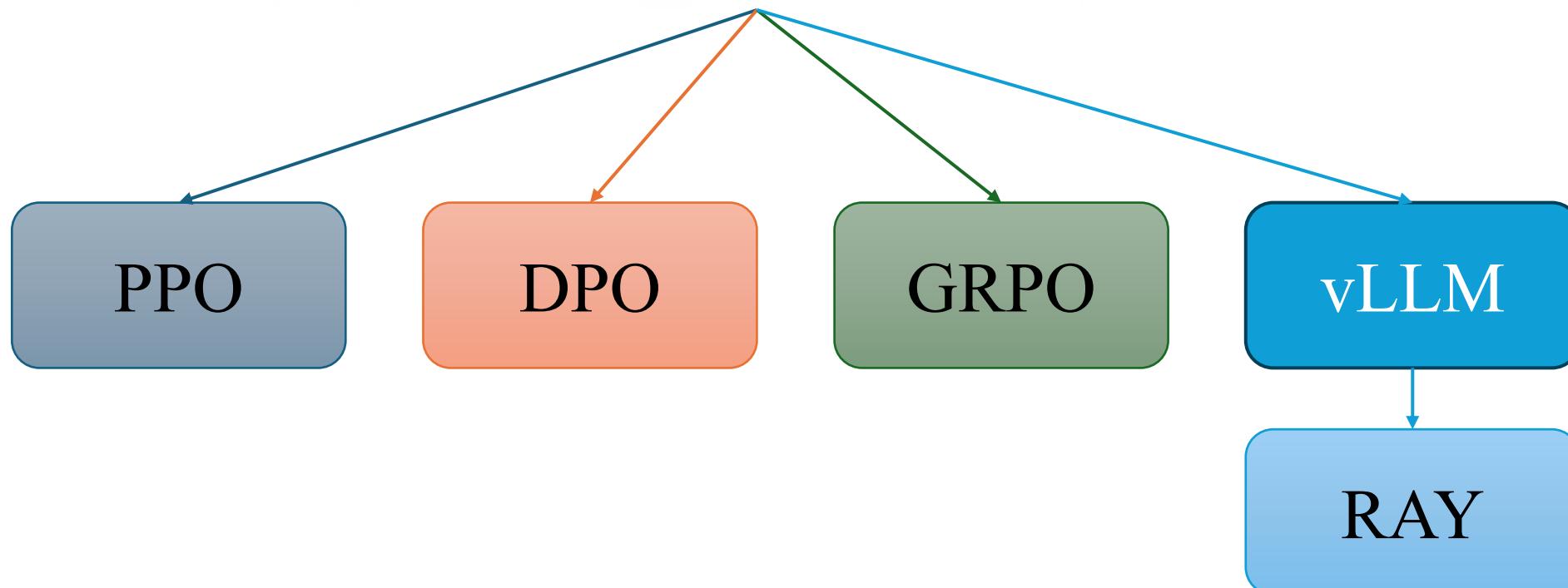
AI VIET NAM
@aivietnam.edu.vn

Reward Model

Reward Model (RM)

❖ OpenRLHF

OPENRLHF





Reward Model (RM)

❖ OpenRLHF: Setup

Installation

To use OpenRLHF, first launch the docker container (**Recommended**) and `pip install openrlhf` inside the docker container:

```
# Launch the docker container
docker run --runtime=nvidia -it --rm --shm-size="10g" --cap-add=SYS_ADMIN -v $PWD:/openrlhf nv
sudo pip uninstall xgboost transformer_engine flash_attn pynvml -y

# pip install
pip install openrlhf

# If you want to use vLLM acceleration (Install vLLM 0.8.3)
pip install openrlhf[vllm]
# latest vLLM is also supported
pip install openrlhf[vllm_latest]
# Install vLLM, ring-flash-attention and Liger-Kernel
pip install openrlhf[vllm,ring,liger]

# pip install the latest version
pip install git+https://github.com/OpenRLHF/OpenRLHF.git

# Or git clone
git clone https://github.com/OpenRLHF/OpenRLHF.git
cd OpenRLHF
pip install -e .
```

The screenshot shows a GitHub repository page for 'OpenRLHF / OpenRLHF'. The repository has 224 issues, 15 pull requests, and 15 discussions. The 'Code' tab is selected, showing the contents of the 'train_rm.py' file. The file contains Python code for training a Reward Model Trainer using DeepSpeed auto TP. It imports argparse, math, os, and datetime from the standard library, along with transformers.trainer, openrlhf.datasets, openrlhf.models, openrlhf.trainer, and openrlhf.utils. The code defines a 'train' function that configures a strategy, sets up distributed training, configures a model, loads a HuggingFace model/config, and initializes a RewardDataset. The 'train_rm.py' file is highlighted in the code editor.

```
import argparse
import math
import os
from datetime import datetime
from transformers.trainer import get_scheduler
from openrlhf.datasets import RewardDataset
from openrlhf.models import get_llm_for_sequence_regression
from openrlhf.trainer import RewardModelTrainer
from openrlhf.utils import blending_datasets, get_strategy, get_tokenizer

def train(args):
    # configure strategy
    strategy = get_strategy(args)
    strategy.setup_distributed()

    # configure model
    # load huggingface model/config
    model = get_llm_for_sequence_regression(
        args.pretrain,
        "reward",
        use_flash_attention_2=args.flash_attn,
        bf16=args.bf16,
        load_in_4bit=args.load_in_4bit,
        lora_rank=args.lora_rank,
        lora_alpha=args.lora_alpha,
        target_modules=args.target_modules,
        lora_dropout=args.lora_dropout,
        ds_config=strategy.get_ds_train_config(is_actor=False),
        init_value_head=True,
        value_head_prefix=args.value_head_prefix,
        packing_samples=args.packing_samples,
    )
```

Reward Model

❖ OpenRLHF: Setup

SFT Model

Dataset

LoRA



```
1 !deepspeed --module openrlhf.cli.train_rm \
2   --save_path ./checkpoint/Llama-3.2-1B-rm-dpo \
3   --save_steps -1 \
4   --logging_steps 1 \
5   --eval_steps -1 \
6   --train_batch_size 96 \
7   --micro_train_batch_size 8 \
8   --pretrain thuanan/Llama-3.2-1B-Instruct-Chat-sft \
9   --value_head_prefix score \
10  --bf16 \
11  --max_epochs 1 \
12  --max_len 2048 \
13  --zero_stage 2 \
14  --learning_rate 5e-6 \
15  --dataset thuanan/Vi-Alpaca-Preference \
16  --apply_chat_template \
17  --chosen_key chosen \
18  --rejected_key rejected \
19  --flash_attn \
20  --load_checkpoint \
21  --packing_samples \
22  --gradient_checkpointing \
23  --adam_offload \
24  --lora_rank 16 \
25  --lora_alpha 32
```

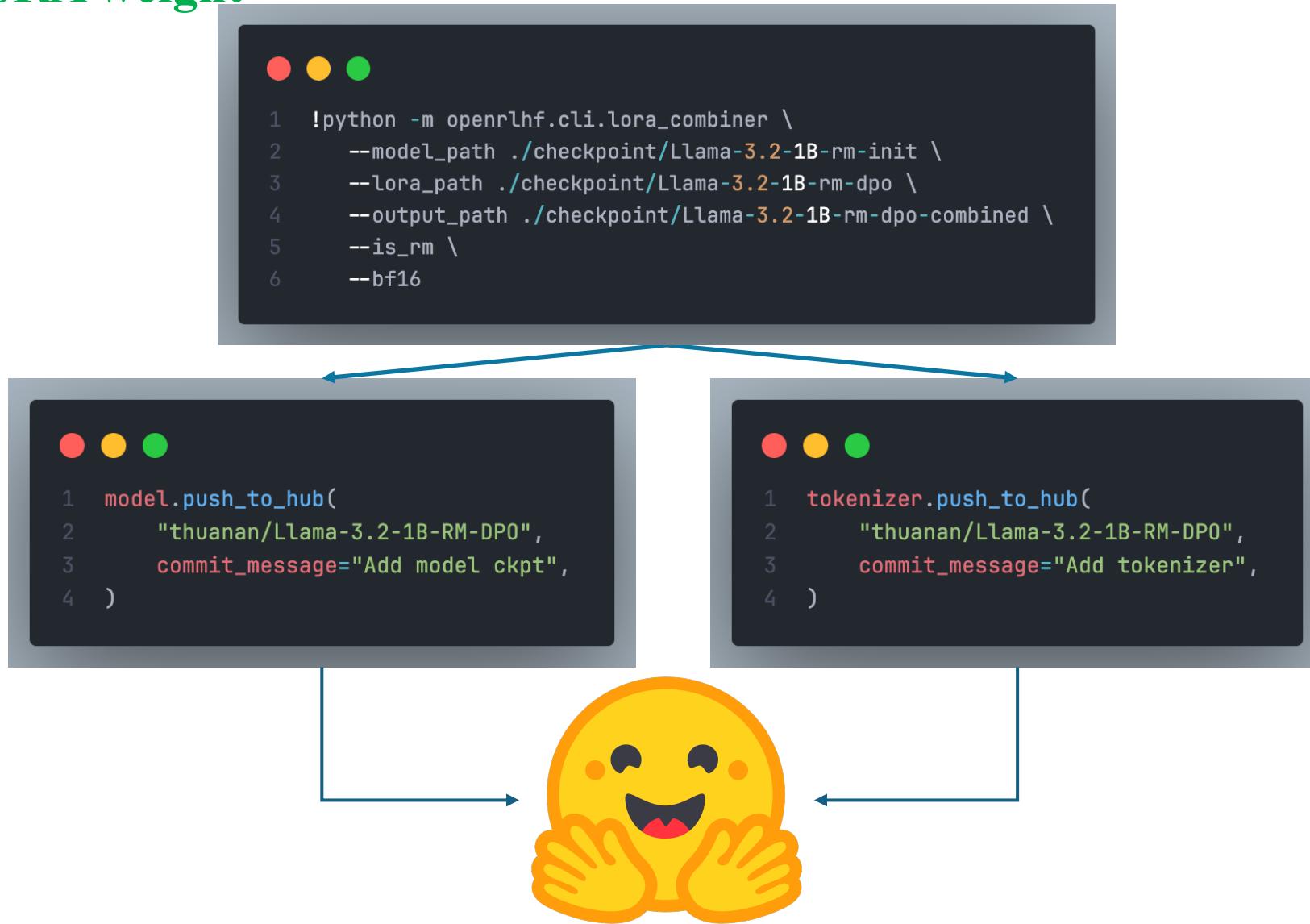


❖ OpenRLHF: Training

```
Train step of epoch 0: 42%|██████████| 1149/2709 [13:32<19:02,  1.37it/s, loss=0.245, acc=0.958, chosen_reward=0.917, reject_reward=-0.658, lr=3.41e-6 [20
25-05-05 01:55:51,191] [INFO] [logging.py:107:log_dist] [Rank 0] step=300, skipped=0, lr=[3.2731563164976977e-06], mom=[(0.9, 0.95)] | 1199/2709 [14:09<18:12,  1.38it/s, loss=0.162, acc=0.958, chosen_reward=1.07, reject_reward=-0.973, lr=3.28e-6]
[2025-05-05 01:55:51,192] [INFO] [timer.py:264:stop] epoch=0/micro_step=1200/global_step=300, RunningAvgSamplesPerSec=34.71104666222326, CurrSamplesPerSec=36.608741413563145, MemAllocated=2.376B, MaxMemAllocated=4.896B
[20
25-05-05 02:00:32,007] [INFO] [logging.py:107:log_dist] [Rank 0] step=400, skipped=0, lr=[2.205908075934387e-06], mom=[(0.9, 0.95)] | 1599/2709 [18:50<11:47,  1.57it/s, loss=0.295, acc=0.875, chosen_reward=1.28, reject_reward=-0.557, lr=2.22e-6]
[2025-05-05 02:00:32,011] [INFO] [timer.py:264:stop] epoch=0/micro_step=1600/global_step=400, RunningAvgSamplesPerSec=34.78872242525318, CurrSamplesPerSec=39.145703543203446, MemAllocated=2.376B, MaxMemAllocated=4.896B
[20
25-05-05 02:05:14,997] [INFO] [logging.py:107:log_dist] [Rank 0] step=500, skipped=0, lr=[1.2610786299917976e-06], mom=[(0.9, 0.95)] | 1999/2709 [23:32<07:29,  1.58it/s, loss=0.151, acc=1, chosen_reward=1.43, reject_reward=-1.04, lr=1.27e-6]
[2025-05-05 02:05:15,006] [INFO] [timer.py:264:stop] epoch=0/micro_step=2000/global_step=500, RunningAvgSamplesPerSec=34.78250022058394, CurrSamplesPerSec=34.1028045448892, MemAllocated=2.376B, MaxMemAllocated=4.896B
[20
25-05-05 02:10:00,658] [INFO] [logging.py:107:log_dist] [Rank 0] step=600, skipped=0, lr=[6.512513249364445e-07], mom=[(0.9, 0.95)] | 2399/2709 [28:18<03:41,  1.40it/s, loss=0.106, acc=1, chosen_reward=1.49, reject_reward=-1.33, lr=6.55e-7]
[2025-05-05 02:10:00,678] [INFO] [timer.py:264:stop] epoch=0/micro_step=2400/global_step=600, RunningAvgSamplesPerSec=34.7208439593949, CurrSamplesPerSec=32.83030974159611, MemAllocated=2.386B, MaxMemAllocated=5.146B
[20
et reward mean stdh 0: 100%|██████████| 2708/2709 [31:57<00:00,  1.42it/s, loss=0.0864, acc=1, chosen_reward=1.72, reject_reward=-1.47, lr=5e-7]
Eval stage of steps 677: 100%|██████████| 27/27 [00:05<00:00,  5.49it/sh
istgram
(tensor([0.0000, 0.0000, 0.0000, 0.0000, 0.2276, 0.1281, 0.1443, 0.0000, 0.0000, 0.0000]), tensor([-10., -8., -6., -4., -2., 0., 2., 4., 6., 8., 10.]), tensor([0.0000, 0.0000, 0.0000, 0.0000, 0.2276, 0.1281, 0.1443, 0.0000, 0.0000, 0.0000]), tensor([-10., -8., -6., -4., -2., 0., 2., 4., 6., 8., 10.])) | 27/27 [00:05<00:00,  5.49it/s, eval_loss=0.16, acc_mean=0.952, reward_mean=0.258, reward_std=1.69]
Eval stage of steps 677: 100%|██████████| 27/27 [00:05<00:00,  5.19it/s, eval_loss=0.16, acc_mean=0.952, reward_mean=0.258, reward_std=1.69]
Train epoch: 100%|██████████| 1/1 [32:03<00:00, 1923.69s/it]
Train step of epoch 0: 100%|██████████| 2709/2709 [32:03<00:00,  1.41it/s, loss=0.122, acc=0.917, chosen_reward=1.92, reject_reward=-1.05, lr=5e-7]
Save value_head_prefix in config
[rank0]:[W505 02:13:49.007376374 ProcessGroupNCCL.cpp:1496] Warning: WARNING: destroy_process_group() was not called before program exit, which can leak resources. For more info, please see https://pytorch.org/docs/stable/distributed.html#shutdown (function operator())
[rank1]:[W505 02:13:49.014926815 ProcessGroupNCCL.cpp:1496] Warning: WARNING: destroy_process_group() was not called before program exit, which can leak resources. For more info, please see https://pytorch.org/docs/stable/distributed.html#shutdown (function operator())
[rank2]:[W505 02:13:49.367378512 ProcessGroupNCCL.cpp:1496] Warning: WARNING: destroy_process_group() was not called before program exit, which can leak resources. For more info, please see https://pytorch.org/docs/stable/distributed.html#shutdown (function operator())
[2025-05-05 02:13:50,722] [INFO] [launch.py:351:main] Process 593113 exits successfully.
[2025-05-05 02:13:50,722] [INFO] [launch.py:351:main] Process 593114 exits successfully.
[2025-05-05 02:13:51,723] [INFO] [launch.py:351:main] Process 593115 exits successfully.
```

Reward Model (RM)

❖ Merge LoRA weight





Training LLMs with RLHF

❖ Push to hub

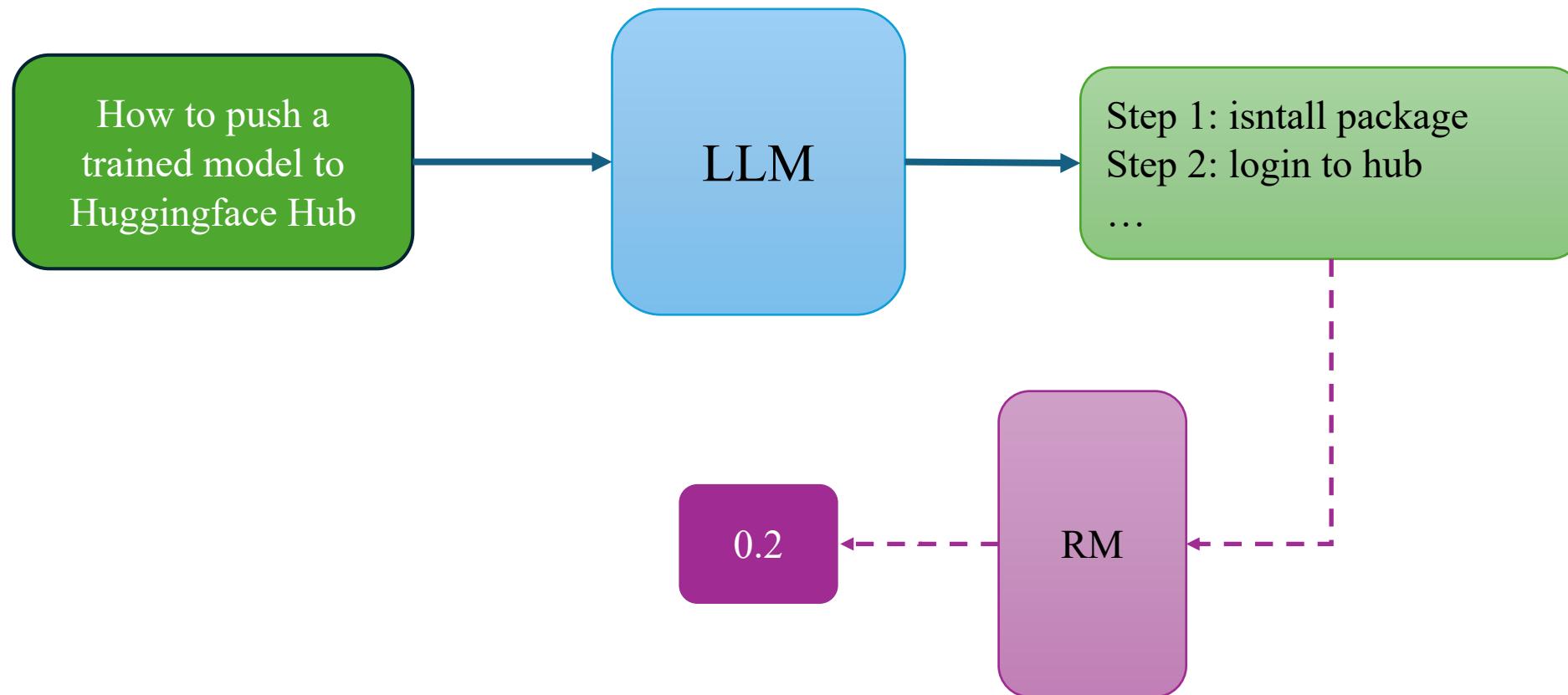
The screenshot shows the Hugging Face Model Hub interface for the Llama-3.2-1B-RM-DPO model. At the top, there's a navigation bar with the repository name "thuanan/Llama-3.2-1B-RM-DPO", a "like" button (0), and several tags: Text Classification, Transformers, Safetensors, llama, text-generation-inference, and arxiv:1910.09700. Below the navigation bar, there are tabs: Model card (selected), Files and versions (the current view), Community, and Settings.

The main content area displays the "Files and versions" tab for the "main" branch. It lists the following files:

File	Size	LFS	Action
.gitattributes	1.57 kB		Add tokenizer
README.md	5.17 kB		Add model ckpt
config.json	1.01 kB		Add model ckpt
model.safetensors	4.94 GB	LFS	Add model ckpt
special_tokens_map.json	454 Bytes		Add tokenizer
tokenizer.json	17.2 MB	LFS	Add tokenizer
tokenizer_config.json	54.7 kB		Add tokenizer

Reward Model (RM)

❖ Inference RM





Training LLMs with RLHF

❖ Inference RM



```
1 inputs = tokenizer(  
2     "Tại sao bạn lại thích học lập trình?",  
3     return_tensors="pt",  
4     max_length=2048,  
5     truncation=True,  
6 )
```

```
model  
[27]  
... LlamaForSequenceClassification(  
    (model): LlamaModel(  
        (embed_tokens): Embedding(128256, 2048, padding_idx=128004)  
        (layers): ModuleList(  
            (0-15): 16 x LlamaDecoderLayer(  
                (self_attn): LlamaAttention(  
                    (q_proj): Linear(in_features=2048, out_features=2048, bias=False)  
                    (k_proj): Linear(in_features=2048, out_features=512, bias=False)  
                    (v_proj): Linear(in_features=2048, out_features=512, bias=False)  
                    (o_proj): Linear(in_features=2048, out_features=2048, bias=False)  
                )  
                (mlp): LlamaMLP(  
                    (gate_proj): Linear(in_features=2048, out_features=8192, bias=False)  
                    (up_proj): Linear(in_features=2048, out_features=8192, bias=False)  
                    (down_proj): Linear(in_features=8192, out_features=2048, bias=False)  
                    (act_fn): SiLU()  
                )  
                (input_layernorm): LlamaRMSNorm((2048,), eps=1e-05)  
                (post_attention_layernorm): LlamaRMSNorm((2048,), eps=1e-05)  
            )  
        )  
        (norm): LlamaRMSNorm((2048,), eps=1e-05)  
        (rotary_emb): LlamaRotaryEmbedding()  
    )  
    (score): Linear(in_features=2048, out_features=1, bias=False)  
)
```



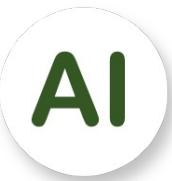
Training LLMs with RLHF

❖ Inference RM

```
reward = model.model(**inputs).last_hidden_state
reward

tensor([[[ 2.3110,  3.2832,  0.0715,  ..., -0.8754, -2.5920,  1.2897],
        [ 0.1436, -0.1955,  0.9064,  ..., -1.4842, -3.2248, -0.3736],
        [ 0.1863,  3.4998,  1.6896,  ...,  1.3823, -2.3932,  1.7603],
        ...,
        [-1.9774,  4.3347, -0.2467,  ...,  1.8927,  1.4284,  2.0450],
        [-1.4019,  2.4985,  1.7585,  ..., -1.9504, -1.1227,  2.3001],
        [-1.0206,  2.1333,  2.1011,  ..., -1.6172, -1.9382, -1.5404]]],  
grad_fn=<MulBackward0>)
```

```
▷ reward = model.score(reward)[:, -1]
reward
[29]
... tensor([[0.0406]], grad_fn=<SelectBackward0>)
```



AI VIET NAM
@aivietnam.edu.vn

RLHF

❖ Setup: Docker

Installation

To use OpenRLHF, first launch the docker container (**Recommended**) and `pip install openrlhf` inside the docker container:

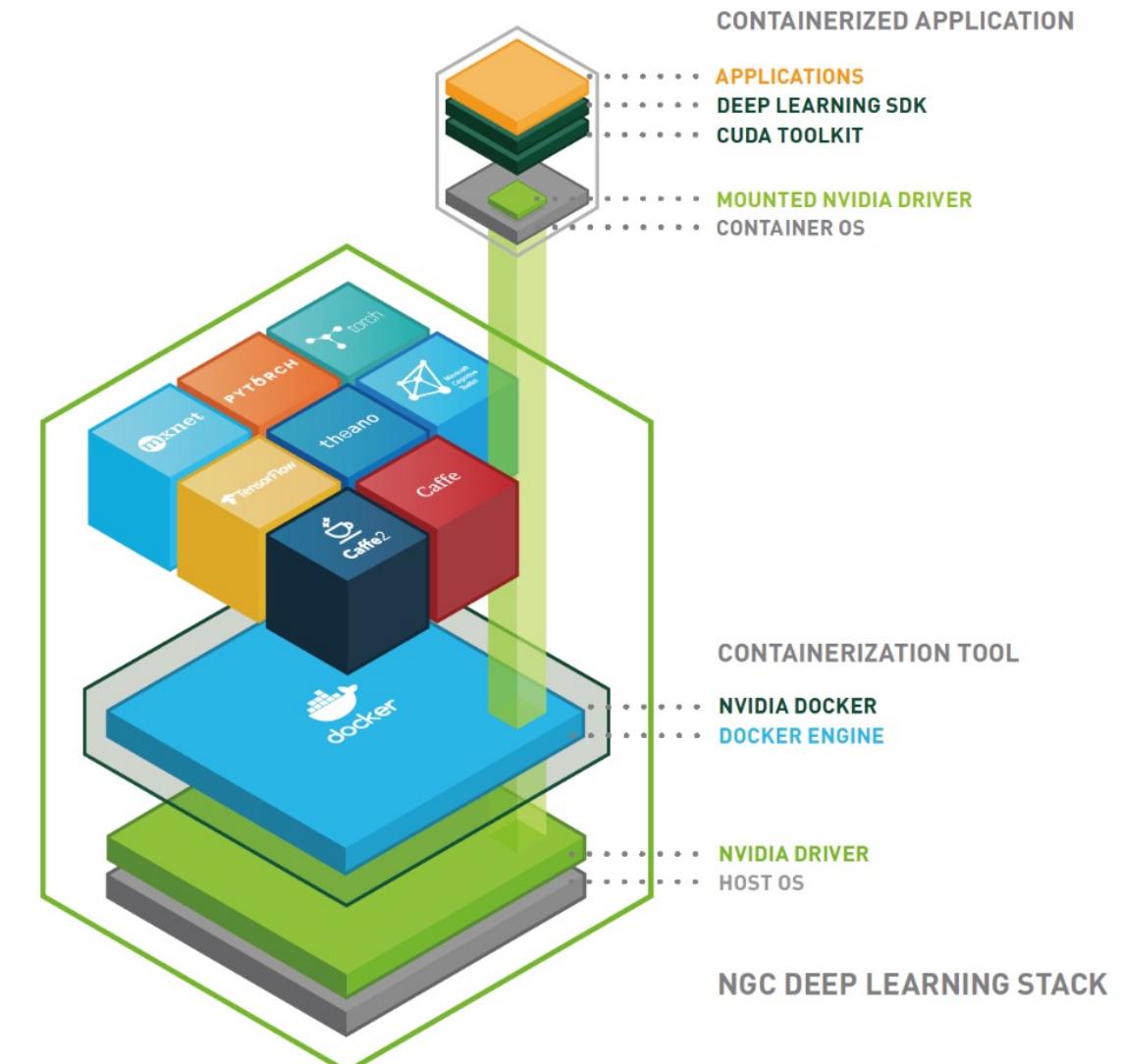
```
# Launch the docker container
docker run --runtime=nvidia -it --rm --shm-size="10g" --cap-add=SYS_ADMIN -v $PWD:/openrlhf nv
sudo pip uninstall xgboost transformer_engine flash_attn pynvml -y

# pip install
pip install openrlhf

# If you want to use vLLM acceleration (Install vLLM 0.8.3)
pip install openrlhf[vllm]
# latest vLLM is also supported
pip install openrlhf[vllm_latest]
# Install vLLM, ring-flash-attention and Liger-Kernel
pip install openrlhf[vllm,ring,liger]

# pip install the latest version
pip install git+https://github.com/OpenRLHF/OpenRLHF.git

# Or git clone
git clone https://github.com/OpenRLHF/OpenRLHF.git
cd OpenRLHF
pip install -e .
```





❖ Setup: Docker

OpenRLHF / examples / scripts / nvidia_docker_install.sh ↗

hijkzzz add flash attention 2 in PPO

Code Blame Executable File · 28 lines (23 loc) · 1.08 KB

```
1 set -x
2
3 # remove old docker
4 sudo apt-get autoremove docker docker-ce docker-engine docker.io containerd runc
5 dpkg -l |grep ^rc|awk '{print $2}' |sudo xargs dpkg --purge
6 sudo apt-get autoremove docker-ce-*
7 sudo rm -rf /etc/systemd/system/docker.service.d
8 sudo rm -rf /var/lib/docker
9
10 # install docker
11 curl https://get.docker.com | sh \
12     && sudo systemctl --now enable docker
13
14 # install nvidia-docker
15 distribution=$(lsb_release -c -s); echo $ID$VERSION_ID) \
16     && curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
17     && curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.list | \
18         sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
19         sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
20
21 sudo apt-get update
22 sudo apt-get install -y nvidia-container-toolkit
23 sudo nvidia-ctk runtime configure --runtime=docker
24
25 sudo groupadd docker
26 sudo usermod -aG docker $USER
27 newgrp docker
28 docker ps
```



❖ Setup: Docker

OpenRLHF / examples / scripts / docker_run.sh

xiaoxigua999 support vllm nccl sync (#543)

Code Blame Executable File · 8 lines (6 loc) · 298 Bytes

```
1 set -x
2
3 PROJECT_PATH=$(cd $(dirname $0)/../../; pwd)
4 IMAGE_NAME="nvcr.io/nvidia/pytorch:24.07-py3"
5
6 docker run --runtime=nvidia -it --rm --shm-size="10g" --cap-add=SYS_ADMIN \
7     -v $PROJECT_PATH:/openrlhf -v $HOME/.cache:/root/.cache -v $HOME/.bash_history2:/root/.bash_history \
8     $IMAGE_NAME bash
```

Installation

To use OpenRLHF, first launch the docker container (**Recommended**) and `pip install openrlhf` inside the docker container:

```
# Launch the docker container
docker run --runtime=nvidia -it --rm --shm-size="10g" --cap-add=SYS_ADMIN -v $PWD:/openrlhf nv
sudo pip uninstall xgboost transformer_engine flash_attn pynvml -y

# pip install
pip install openrlhf

# If you want to use VLLM acceleration (Install vLLM 0.8.3)
pip install openrlhf[vllm]
# latest vLLM is also supported
pip install openrlhf[vllm_latest]
# Install vLLM, ring-flash-attention and Liger-Kernel
pip install openrlhf[vllm,ring,liger]
```

PPO RLHF

❖ Setup: RAY

```
ray start --head --node-ip-address 0.0.0.0 --num-gpus 3
```

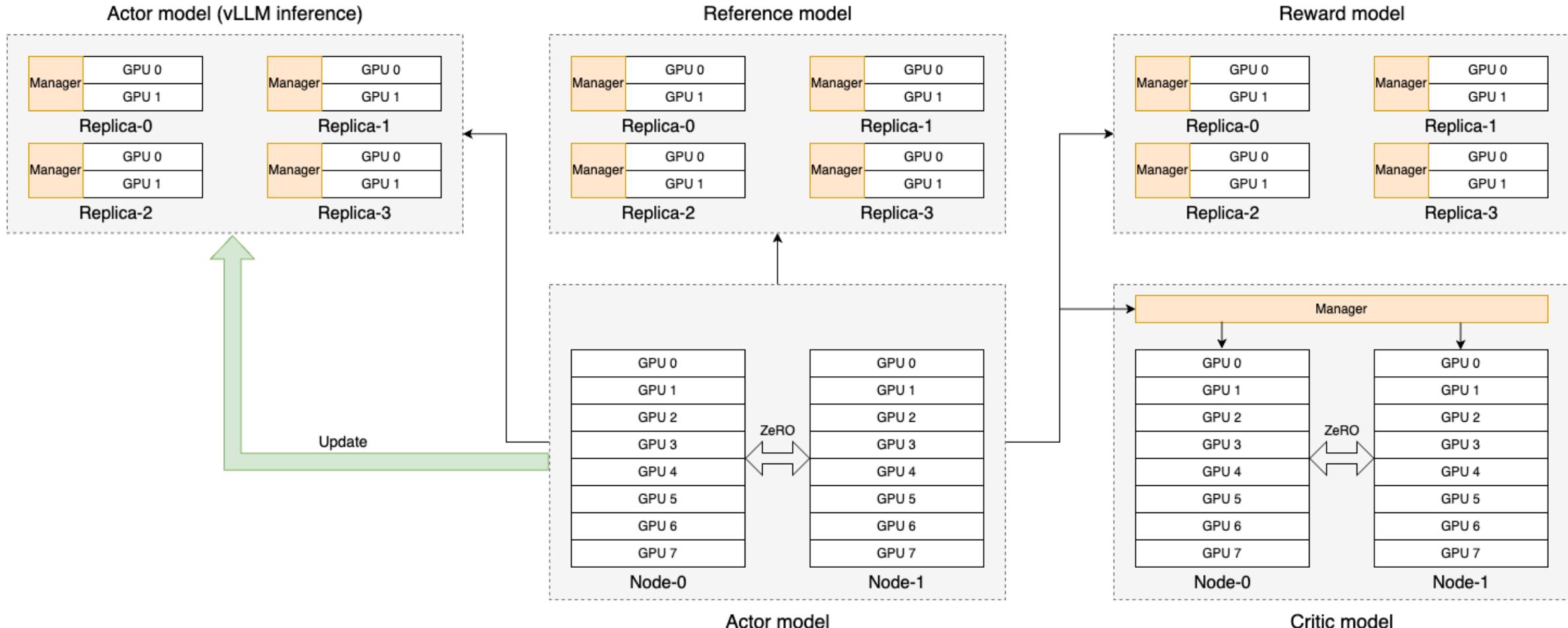
```
===== Autoscaler status: 2025-05-06 23:59:15.077748 =====
Node status
-----
Active:
  1 node_0e76bf9f229cf6ee08248f527533729807f964f6ae654f665fa860c6
Pending:
  (no pending nodes)
Recent failures:
  (no failures)

Resources
-----
Usage:
  0.0/80.0 CPU
  0.0/3.0 GPU
  0B/105.85GiB memory
  0B/9.50GiB object_store_memory

Demands:
  (no resource demands)
```

PPO RLHF

❖ Setup: RAY - Architecture



❖ Setup: Dataset

Datasets: thuanan/Prompt-Vi-Alpaca-Preference-2k		Dataset card
Split (1) train · 2k rows		
<input type="text"/> Search this dataset		
context_messages list · lengths	context string · lengths	
1 100%	10..861 96.6%	
[{ "content": "Dự đoán đầu ra của câu lệnh JavaScript này\n\nvà a = 3;\nbien b = 6;\nconsole.log(a+b)", "role": "user" }]	Dự đoán đầu ra của câu lệnh JavaScript này và a = 3; biến b = 6; console.log(a+b)	
[{ "content": "Soạn thảo một e-mail đến một khu nghỉ dưỡng để hỏi về thông tin liên quan đến một lễ kỷ niệm đổi mới lời thề cho vợ tôi và tôi.", "role": "user" }]	Soạn thảo một e-mail đến một khu nghỉ dưỡng để hỏi về thông tin liên quan đến một lễ kỷ niệm đổi mới lời thề cho vợ tôi và tôi.	
[{ "content": "Tạo ý tưởng sản phẩm mới liên quan đến bao bì thực phẩm.", "role": "user" }]	Tạo ý tưởng sản phẩm mới liên quan đến bao bì thực phẩm.	
[{ "content": "Đua xe đạp Tour De France là gì?", "role": "user" }]	Đua xe đạp Tour De France là cuộc đua xe đạp hàng năm tại Pháp. Cuộc đua này diễn ra hàng năm và thu hút hàng nghìn vận động viên từ khắp nơi trên thế giới.	
[{ "content": "Tạo câu có mệnh đề quan hệ", "role": "user" }]	Tạo câu có mệnh đề quan hệ	
[{ "content": "Những nước nào nói tiếng Tây Ban Nha?", "role": "user" }]	Những nước nào nói tiếng Tây Ban Nha?	
[{ "content": "Chung kết ATP World Tour là gì?", "role": "user" }]	Chung kết ATP World Tour là gì?	
[{ "content": "Viết một vài câu về một nhân vật lịch sử nhất định.\n\nAmelia Earhart", "role": "user" }]	Viết một vài câu về một nhân vật lịch sử nhất định. Amelia Earhart	
[{ "content": "Thách thức môi trường quan trọng nhất mà nhân loại hiện đang phải đối mặt là gì?", "role": "user" }]	Thách thức môi trường quan trọng nhất mà nhân loại hiện đang phải đối mặt là gì?	
[{ "content": "Tại sao rất nhiều người sống dọc theo bờ biển?", "role": "user" }]	Tại sao rất nhiều người sống dọc theo bờ biển?	
[{ "content": "tại sao nhiều phụ nữ làm việc có tổ chức và có phương pháp hơn so với nam giới", "role": "user" }]	tại sao nhiều phụ nữ làm việc có tổ chức và có phương pháp hơn so với nam giới	



PPO RLHF

❖ Setup: Args

```
OpenRLHF / openrlhf / cli / train_ppo_ray.py

xiaoxygu999 Support deepspeed auto tp (#996) ... ✓

Code Blame 474 lines (421 loc) · 21.2 KB

1 import argparse
2 from datetime import datetime
3
4 import ray
5 from ray.util.placement_group import placement_group
6
7 from openrlhf.trainer.ppo_trainer import PPOTrainer
8 from openrlhf.trainer.ray import (
9     PPORayActorGroup,
10    ReferenceModelRayActor,
11    RewardModelRayActor,
12    create_vllm_engines,
13 )
14 from openrlhf.trainer.ray.ppo_actor import ActorModelRayActor
15 from openrlhf.trainer.ray.ppo_critic import CriticModelRayActor
16 from openrlhf.utils import get_strategy
17
18
19 def train(args):
20     # configure strategy
21     strategy = get_strategy(args)
22     strategy.print(args)
23
24     # init vllm / actor /critic /ref /reward model
25     # if colocated, create placement group for actor and ref model explicitly.
26     pg = None
27     if args.colocate_actor_ref or args.colocate_all_models:
28         if args.init_kl_coef > 0:
29             assert (
30                 args.actor_num_nodes == args.ref_num_nodes
31                 and args.actor_num_gpus_per_node == args.ref_num_gpus_per_node
32             ), f"num_nodes and num_gpus_per_node must be the same when colocate actor and ref model."
33
34     bundles = [{"GPU": 1, "CPU": 1} for _ in range(args.actor_num_nodes * args.actor_num_gpus_per_node)]
35     pg = placement_group(bundles, strategy="PACK")
36     ray.get(pg.ready())
37
38
39
40
41
```

```
ray job submit --address="http://127.0.0.1:8265" \
--runtime-env-json='{"setup_commands": ["pip install openrlhf[vllm]"]}' \
-- python3 -m openrlhf.cli.train_ppo_ray \
--ref_num_nodes 1 \
--ref_num_gpus_per_node 1 \
--reward_num_nodes 1 \
--reward_num_gpus_per_node 1 \
--critic_num_nodes 1 \
--critic_num_gpus_per_node 1 \
--actor_num_nodes 1 \
--actor_num_gpus_per_node 1 \
--vllm_num_engines 1 \
--vllm_tensor_parallel_size 1 \
--colocate_critic_reward \
--colocate_actor_ref \
--pretrain thuanan/Llama-3.2-1B-Instruct-Chat-sft \
--reward_pretrain thuanan/Llama-3.2-1B-RM-DPO \
--save_path /openrlhf/examples/checkpoint/Llama-3.2-1B-RLHF-2k \
--micro_train_batch_size 4 \
--train_batch_size 64 \
--micro_rollout_batch_size 8 \
--rollout_batch_size 512 \
--max_epochs 2 \
--prompt_max_len 1024 \
--generate_max_len 1024 \
--zero_stage 3 \
--bf16 \
--actor_learning_rate 5e-7 \
--critic_learning_rate 9e-6 \
--init_kl_coef 0.01 \
--prompt_data thuanan/Prompt-Vi-Alpaca-Preference-2k \
--input_key context_messages \
--apply_chat_template \
--normalize_reward \
--packing_samples \
--adam_offload \
--flash_attn \
--gradient_checkpointing \
--load_checkpoint \
--use_wandb ec64e817474d056ee6ef4cc78513c453cffaf34
```



❖ Training process

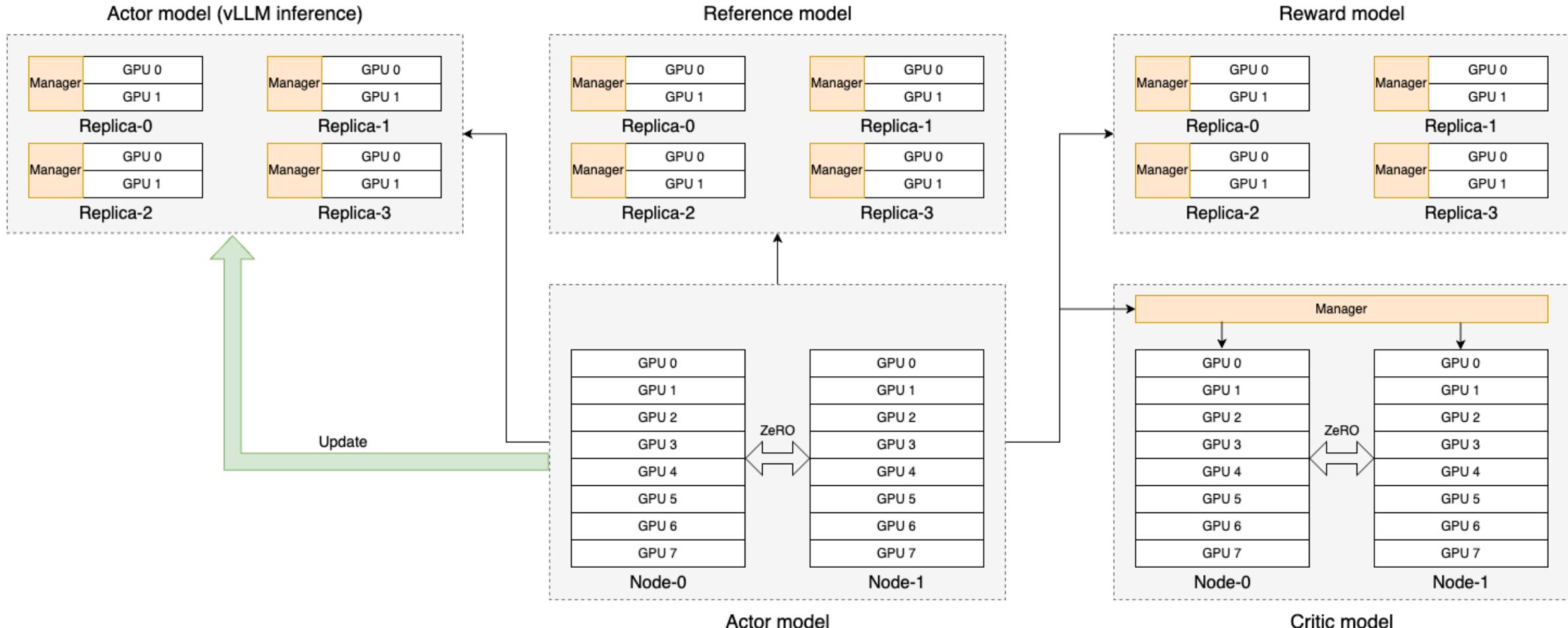
```
Train epoch [1/1]: 98% | 251/256 [06:18<00:04, 1.00it/s, act_loss=-0.0767, reward=2.65, return=2.64, gen_len=502, tot_len=655, kl=0.00281, act_lr=3.55e-7]
Train epoch [1/1]: 98% | 252/256 [06:18<00:03, 1.00it/s, act_loss=-0.0767, reward=2.65, return=2.64, gen_len=502, tot_len=655, kl=0.00281, act_lr=3.55e-7]
(ActorModelRayActor pid=42197)
Train epoch [1/1]: 98% | 252/256 [06:19<00:03, 1.00it/s, act_loss=0.435, reward=2.19, return=2.2, gen_len=658, tot_len=777, kl=-0.0023, act_lr=3.55e-7]
Train epoch [1/1]: 99% | 253/256 [06:19<00:03, 1.01s/it, act_loss=0.435, reward=2.19, return=2.2, gen_len=658, tot_len=777, kl=-0.0023, act_lr=3.55e-7]
(ActorModelRayActor pid=42197)
Train epoch [1/1]: 99% | 253/256 [06:20<00:03, 1.01s/it, act_loss=0.319, reward=2.65, return=2.65, gen_len=387, tot_len=569, kl=0.00546, act_lr=3.55e-7]
Train epoch [1/1]: 99% | 254/256 [06:20<00:01, 1.01it/s, act_loss=0.319, reward=2.65, return=2.65, gen_len=387, tot_len=569, kl=0.00546, act_lr=3.55e-7]
(ActorModelRayActor pid=42197)
Train epoch [1/1]: 99% | 254/256 [06:21<00:01, 1.01it/s, act_loss=-0.645, reward=3.03, return=3.03, gen_len=410, tot_len=531, kl=3.29e-5, act_lr=3.55e-7]
Train epoch [1/1]: 100% | 255/256 [06:21<00:00, 1.02it/s, act_loss=-0.645, reward=3.03, return=3.03, gen_len=410, tot_len=531, kl=3.29e-5, act_lr=3.55e-7]
(ActorModelRayActor pid=42197)
Train epoch [1/1]: 100% | 255/256 [06:30<00:00, 1.02it/s, act_loss=0.576, reward=2.53, return=2.53, gen_len=487, tot_len=586, kl=-0.000108, act_lr=3.51e-7]
Train epoch [1/1]: 100% | 256/256 [06:30<00:00, 3.25s/it, act_loss=0.576, reward=2.53, return=2.53, gen_len=487, tot_len=586, kl=-0.000108, act_lr=3.51e-7]
Train epoch [1/1]: 100% | 256/256 [06:30<00:00, 1.52s/it, act_loss=0.576, reward=2.53, return=2.53, gen_len=487, tot_len=586, kl=-0.000108, act_lr=3.51e-7]
(LLMRayActor pid=42194) update weight: model.layers.14.mlp.gate_proj.weight, dtype: torch.bfloat16, shape: [8192, 2048]
(LLMRayActor pid=42194) update weight: model.layers.14.mlp.up_proj.weight, dtype: torch.bfloat16, shape: [8192, 2048]
(LLMRayActor pid=42194) update weight: model.layers.14.mlp.down_proj.weight, dtype: torch.bfloat16, shape: [2048, 8192]
(LLMRayActor pid=42194) update weight: model.layers.14.input_layernorm.weight, dtype: torch.bfloat16, shape: [2048]
(LLMRayActor pid=42194) update weight: model.layers.14.post_attention_layernorm.weight, dtype: torch.bfloat16, shape: [2048]
(LLMRayActor pid=42194) update weight: model.layers.15.self_attn.q_proj.weight, dtype: torch.bfloat16, shape: [2048, 2048]
(LLMRayActor pid=42194) update weight: model.layers.15.self_attn.k_proj.weight, dtype: torch.bfloat16, shape: [512, 2048]
(LLMRayActor pid=42194) update weight: model.layers.15.self_attn.v_proj.weight, dtype: torch.bfloat16, shape: [512, 2048]
(LLMRayActor pid=42194) update weight: model.layers.15.self_attn.o_proj.weight, dtype: torch.bfloat16, shape: [2048, 2048]
(LLMRayActor pid=42194) update weight: model.layers.15.mlp.gate_proj.weight, dtype: torch.bfloat16, shape: [8192, 2048]
(LLMRayActor pid=42194) update weight: model.layers.15.mlp.up_proj.weight, dtype: torch.bfloat16, shape: [8192, 2048]
(LLMRayActor pid=42194) update weight: model.layers.15.mlp.down_proj.weight, dtype: torch.bfloat16, shape: [2048, 8192]
(LLMRayActor pid=42194) update weight: model.layers.15.input_layernorm.weight, dtype: torch.bfloat16, shape: [2048]
(LLMRayActor pid=42194) update weight: model.layers.15.post_attention_layernorm.weight, dtype: torch.bfloat16, shape: [2048]
(LLMRayActor pid=42194) update weight: model.norm.weight, dtype: torch.bfloat16, shape: [2048]
(LLMRayActor pid=42194) update weight: model.embed_tokens.weight, dtype: torch.bfloat16, shape: [128256, 2048]
(LLMRayActor pid=42194) update weight: model.layers.0.self_attn.q_proj.weight, dtype: torch.bfloat16, shape: [2048, 2048]
```

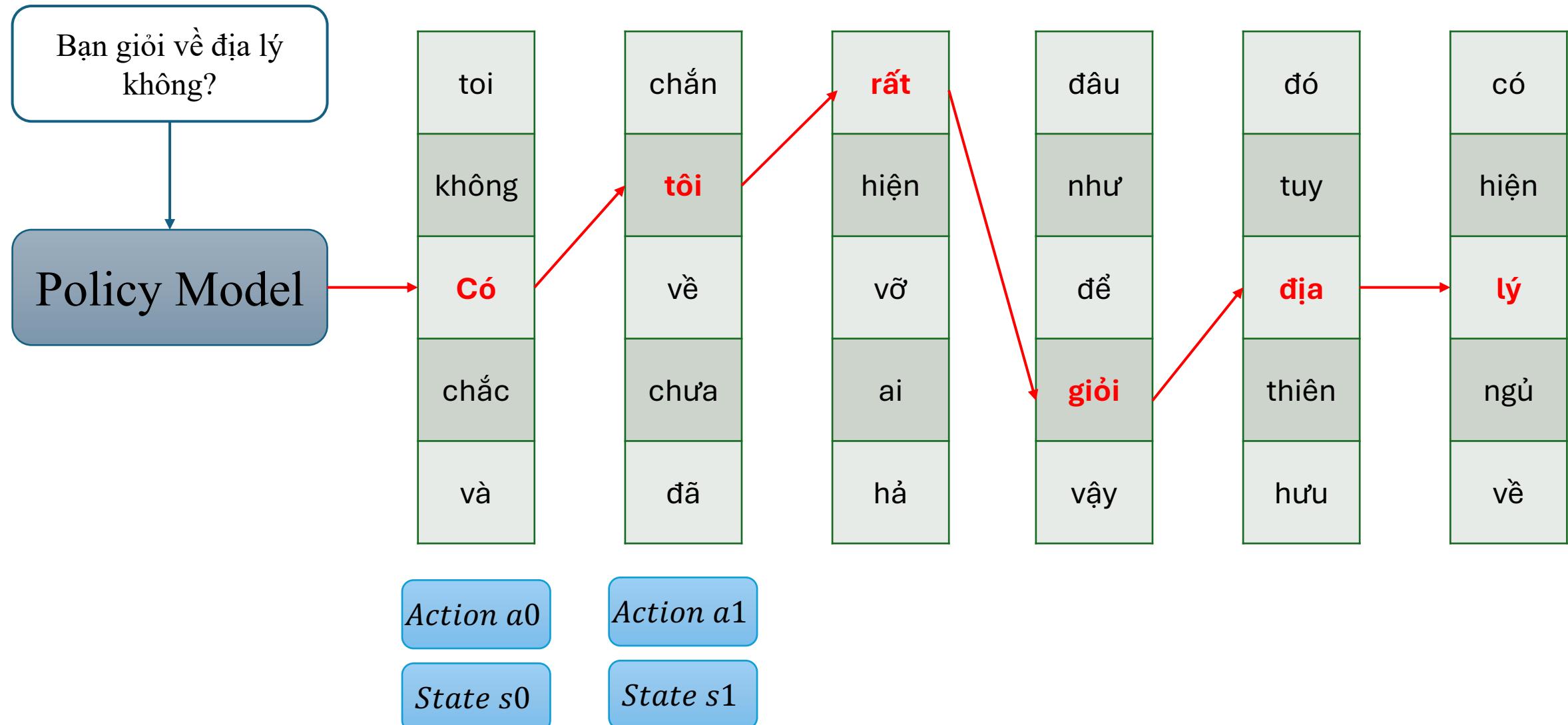


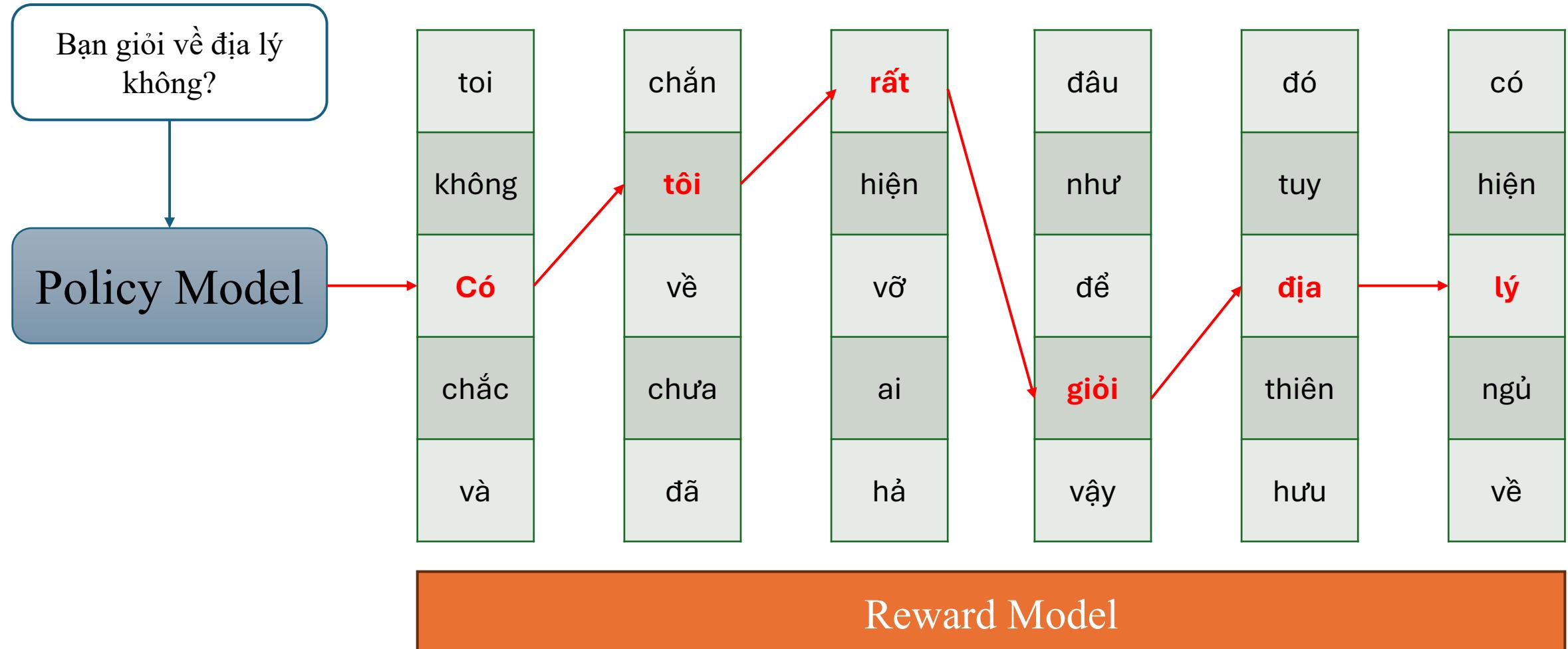
❖ Training process

```
Train epoch [2/2]: 99% | 127/128 [02:42<00:00, 1.09it/s, critic_loss=0.976, values=3.37, critic_lr=9.18e-7]
(CriticModelRayActor pid=75065)
Train epoch [2/2]: 99% | 127/128 [02:49<00:00, 1.09it/s, critic_loss=0.296, values=3.17, critic_lr=9.14e-7]
Train epoch [2/2]: 100% | 128/128 [02:49<00:00, 2.67s/it, critic_loss=0.296, values=3.17, critic_lr=9.14e-7]
Train epoch [2/2]: 100% | 128/128 [02:49<00:00, 1.32s/it, critic_loss=0.296, values=3.17, critic_lr=9.14e-7]
(PPOTrainer pid=75071)
Episode [1/1]: 95% | 18/19 [2:17:14<07:05, 425.44s/it, policy_loss=0.0454, actor_lr=5.35e-8, kl=0.0163, reward=3.03, response_length=230, total_length=369, return=2.99, critic_loss=0.814, values=3.25, critic_lr=9.64e-7]
(PPOTrainer pid=75071)
Episode [1/1]: 100% | 19/19 [2:17:14<00:00, 416.84s/it, policy_loss=0.0454, actor_lr=5.35e-8, kl=0.0163, reward=3.03, response_length=230, total_length=369, return=2.99, critic_loss=0.814, values=3.25, critic_lr=9.64e-7]
(PPOTrainer pid=75071) wandb: uploading artifact run-su6ovgse-traiinggenerated_samples; uploading output.log; uploading wandb-summary.json; uploading config.yaml
(PPOTrainer pid=75071) wandb: uploading artifact run-su6ovgse-traiinggenerated_samples
(PPOTrainer pid=75071) wandb: uploading history steps 36-37, summary
(PPOTrainer pid=75071) wandb:
(PPOTrainer pid=75071) wandb:
(PPOTrainer pid=75071) wandb: Run history:
(PPOTrainer pid=75071) wandb:    train/actor_lr [REDACTED]
(PPOTrainer pid=75071) wandb:    train/critic_loss [REDACTED]
(PPOTrainer pid=75071) wandb:    train/critic_lr [REDACTED]
(PPOTrainer pid=75071) wandb:    train/global_step [REDACTED]
(PPOTrainer pid=75071) wandb:    train/kl [REDACTED]
(PPOTrainer pid=75071) wandb:    train/policy_loss [REDACTED]
(PPOTrainer pid=75071) wandb:    train/response_length [REDACTED]
(PPOTrainer pid=75071) wandb:    train/return [REDACTED]
(PPOTrainer pid=75071) wandb:    train/reward [REDACTED]
(PPOTrainer pid=75071) wandb:    train/total_length [REDACTED]
(PPOTrainer pid=75071) wandb:    train/values [REDACTED]
(PPOTrainer pid=75071) wandb:
(PPOTrainer pid=75071) wandb: Run summary:
(PPOTrainer pid=75071) wandb:    train/actor_lr 0.0
(PPOTrainer pid=75071) wandb:    train/critic_loss 0.81358
(PPOTrainer pid=75071) wandb:    train/critic_lr 0.0
(PPOTrainer pid=75071) wandb:    train/global_step 19
(PPOTrainer pid=75071) wandb:    train/kl 0.01627
(PPOTrainer pid=75071) wandb:    train/policy_loss 0.04543
(PPOTrainer pid=75071) wandb:    train/response_length 230.29102
(PPOTrainer pid=75071) wandb:    train/return 2.98979
(PPOTrainer pid=75071) wandb:    train/reward 3.03121
(PPOTrainer pid=75071) wandb:    train/total_length 369.47656
(PPOTrainer pid=75071) wandb:    train/values 3.24503
(PPOTrainer pid=75071) wandb:
(PPOTrainer pid=75071) wandb: 🎉 View run ppo_050ST20:38 at: https://wandb.ai/thuan-nan-ai-viet-nam/openrlhf_train_ppo/runs/su6ovgse
(PPOTrainer pid=75071) wandb: ★ View project at: https://wandb.ai/thuan-nan-ai-viet-nam/openrlhf_train_ppo
(PPOTrainer pid=75071) wandb: Synced 5 W&B file(s), 19 media file(s), 38 artifact file(s) and 0 other file(s)
(PPOTrainer pid=75071) wandb: Find logs at: ./wandb/run-20250505_203844-su6ovgse/logs
(PPOTrainer pid=75071)
Episode [1/1]: 100% | 19/19 [2:17:19<00:00, 433.65s/it, policy_loss=0.0454, actor_lr=5.35e-8, kl=0.0163, reward=3.03, response_length=230, total_length=369, return=2.99, critic_loss=0.814, values=3.25, critic_lr=9.64e-7]
```

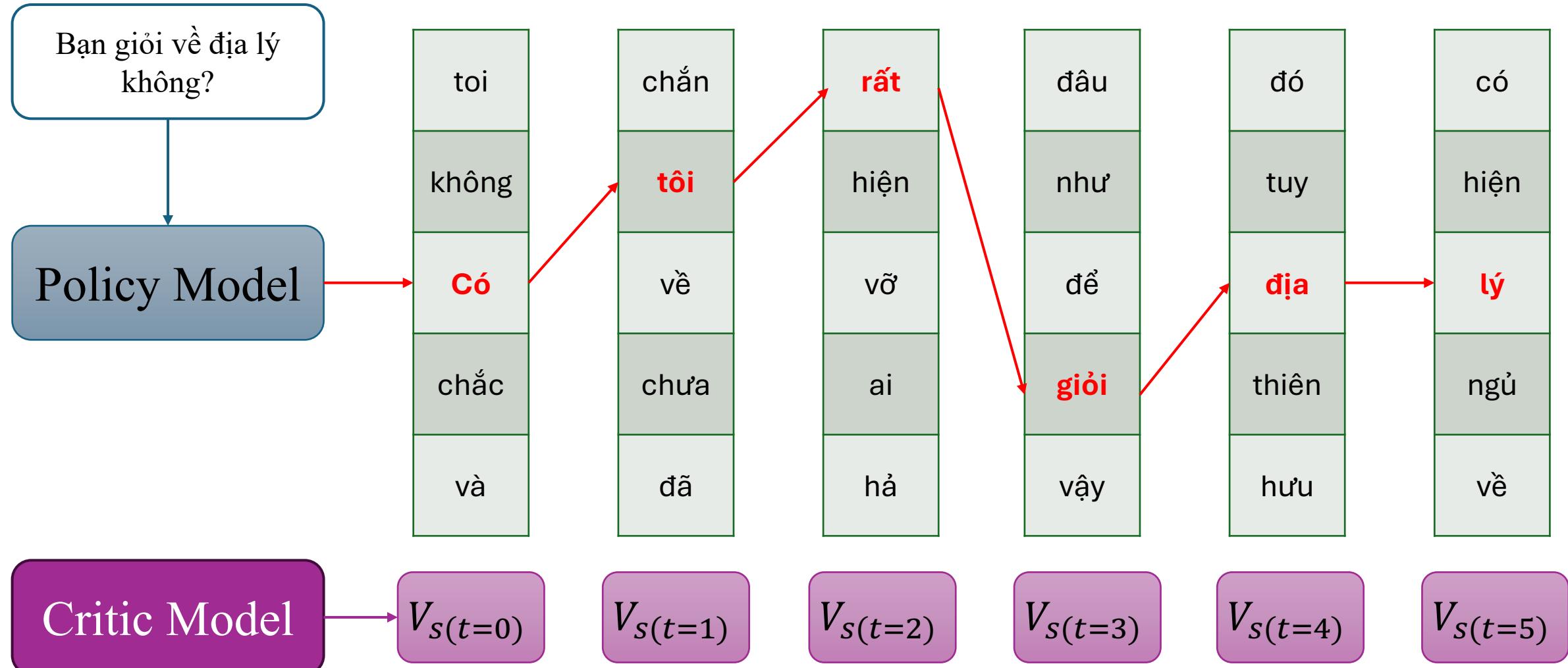
❖ Setup: RAY - Architecture







Score = (2.1)



❖ PPO-clip Loss

$$L_t^{clip}(s, a, \theta) = \min(r_t(\theta)A_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)$$

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_old}(a_t | s_t)}$$

$$A_t = R - V_{S_t}$$

❖ RL

Monte Carlo Approach:

Monte Carlo: waits until the end of the episode, then calculates G_t (return) and uses it as a target for its value or policy.

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

New value of state t

Former estimation
of value of state t
(= Expected return
starting at that state)

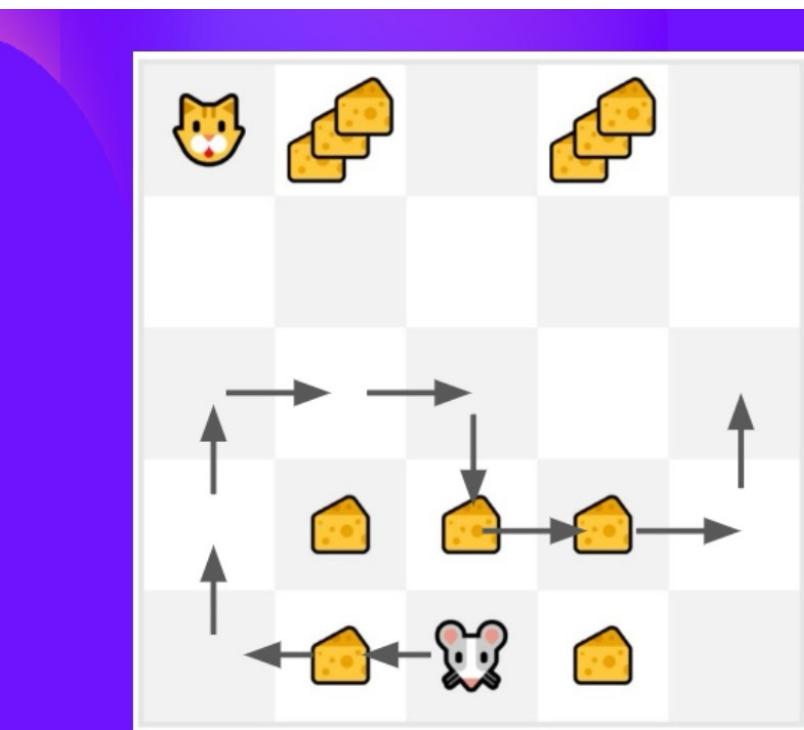
Learning
Rate

Return at
timestep
t

Former estimation
of value of state t
(= Expected return
starting at that state)

❖ RL

- Initialize value function to zero for each state
- Learning rate (lr) is 0.1 and our discount rate is 1 (= no discount)
- The mouse explores the environment and takes random actions



- Calculate the return G_t .
 $G_t = R_{t+1} + R_{t+2} + R_{t+3} \dots$
 $G_t = 1 + 0 + 0 + 0 + 0 + 0 + 1 + 1 + 0 + 0$
 $G_t = 3$

- We can now update $V(S_0)$.

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

$$\begin{aligned} \text{New } V(S_0) &= V(S_0) + lr * [G_t - V(S_0)] \\ \text{New } V(S_0) &= 0 + 0.1 * [3 - 0] \\ \text{New } V(S_0) &= 0.3 \end{aligned}$$

❖ RL

TD Learning Approach:

Temporal Difference Learning: learning at each time step.

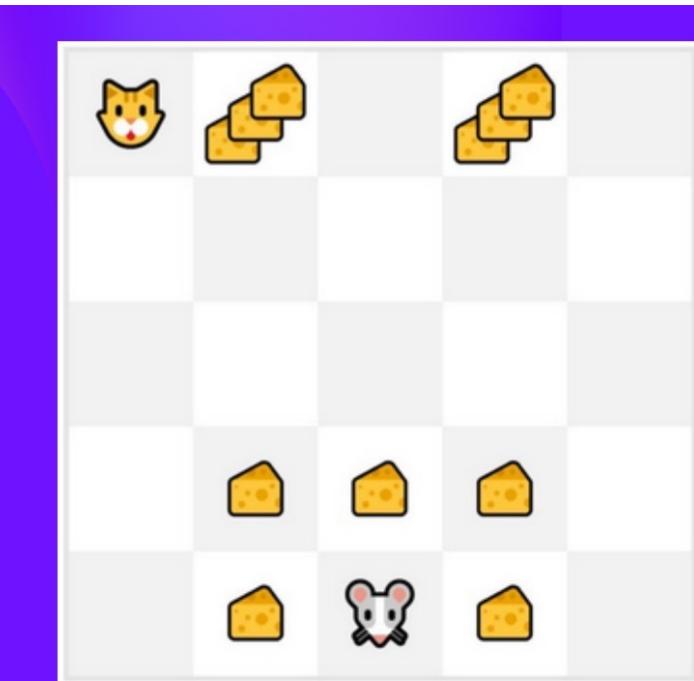
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

New value of state t Former estimation of value of state t Learning Rate Reward Discounted value of next state

TD Target

❖ RL

- Initialize value function to zero for each state
- Learning rate (lr) is 0.1 and our discount rate is 1 (= no discount)
- The mouse explores the environment and takes random actions (**going to the left**)



At the end of one step (State, Action, Reward, Next State):

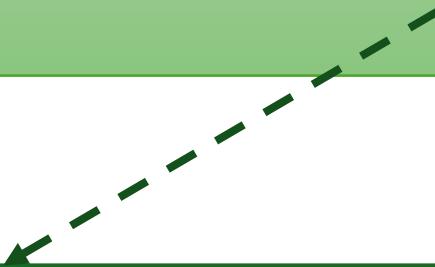
- We have R_{t+1} and S_{t+1}
- We update $V(S_t)$:
 - We estimate G_t by adding R_{t+1} and the discounted value of next state.
TD target : $R_{t+1} + \gamma V(S_{t+1})$

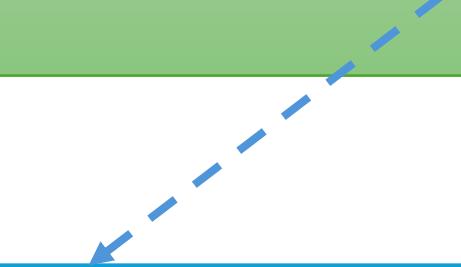
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Now we continue to interact with this environment with our updated value function. By running more and more steps, the agent will learn to play better and better.

❖ PPO-clip Loss

$$L_t^{clip}(s, a, \theta) = \min(r_t(\theta)A_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)$$


$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_old}(a_t | s_t)}$$


$$A_t = R - V_{S_t}$$

$$A_t = \sum_{l=0}^{T-t} (\gamma \lambda)^l [r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l})]$$

❖ PPO-clip Loss

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_old}(a_t | s_t)}$$

*

$$A_t = R - V_{S_t}$$

If $A(t) > 0$: the action was better than expected → increase its probability.

If $A(t) < 0$: the action was worse than expected → decrease its probability.

$$\mathcal{L} = \mathcal{L}_{PPO} + \beta * KL(\pi_{new}, \pi_{ref})$$

PPO RLHF

❖ Training process





PPO RLHF

❖ Push to Hub

The screenshot shows two code snippets demonstrating how to push a model and its tokenizer to the Hugging Face Model Hub.

Left Panel (Model Push):

```
1 model.push_to_hub(  
2     "thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca",  
3     commit_message="Add model ckpt",  
4 )
```

Right Panel (Tokenizer Push):

```
1 tokenizer.push_to_hub(  
2     "thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca",  
3     commit_message="Add tokenizer",  
4 )
```

Below the code snippets is a screenshot of the Hugging Face Model Hub interface for the repository `thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca`.

Repository Details:

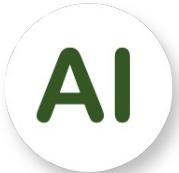
- Owner: thuanan
- Name: Llama-3.2-1B-RLHF-2k-vi-alpaca
- Last commit: Add tokenizer ckpt (15d4ed5)
- Verified

Files and Versions:

- main (selected)
- Llama-3.2-1B-RLHF-2k-vi-alpaca

File List:

File	Size	Action
.gitattributes	1.57 kB	Add tokenizer ckpt
README.md	5.17 kB	Add model ckpt
config.json	922 Bytes	Add model ckpt
generation_config.json	234 Bytes	Add model ckpt
model.safetensors	4.94 GB	LFS Add model ckpt
special_tokens_map.json	454 Bytes	Add tokenizer ckpt
tokenizer.json	17.2 MB	LFS Add tokenizer ckpt
tokenizer_config.json	54.7 kB	Add tokenizer ckpt



AI

AI VIET NAM
@aivietnam.edu.vn

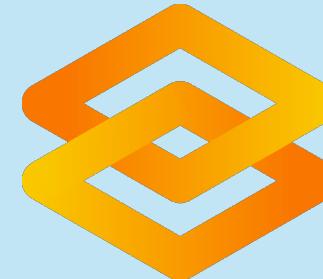
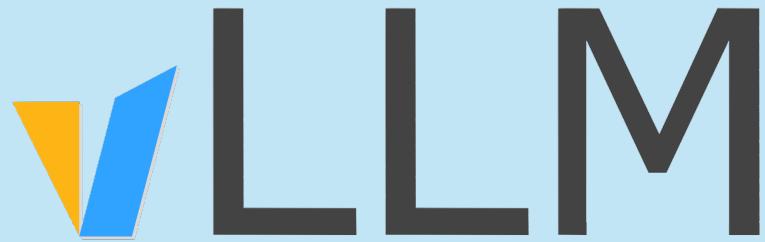
QUIZ



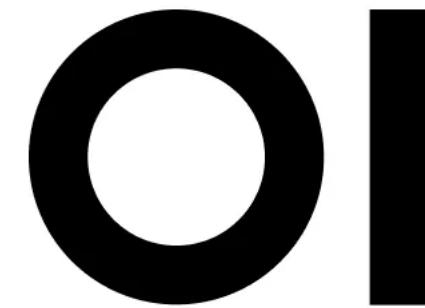
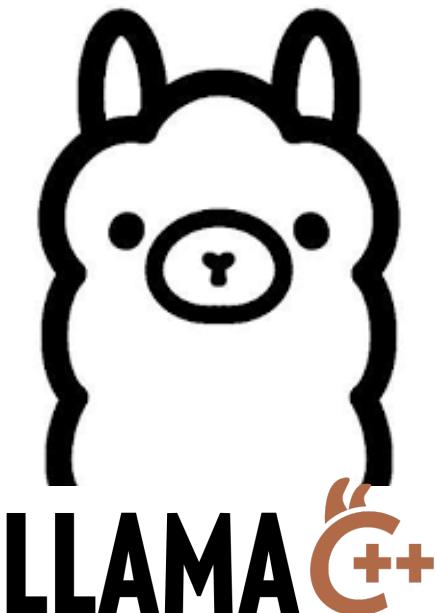
Building Chat Interface

Building Chat Interface

❖ Getting Started



gradio



Open-webui

Building Chat Interface

❖ vLLM with Gradio

The image shows a screenshot of a GitHub repository page for 'thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca'. The repository has 0 likes. The navigation bar includes 'Text Generation', 'Transformers', 'Safetensors', 'llama', 'conversational', 'text-generation-inference', and a link to 'arxiv:1910.09700'. Below the navigation bar, there are tabs for 'Model card', 'Files and versions' (which is selected), 'Community', and 'Settings'. The 'Files and versions' tab shows the following files:

- thuanan Add tokenizer ckpt 15d4ed5 (VERIFIED)
- .gitattributes
- README.md
- config.json
- generation_config.json
- model.safetensors (4.94 GB, LFS)
- special_tokens_map.json
- tokenizer.json (17.2 MB, LFS)
- tokenizer_config.json (54.7 kB)

An orange arrow points from the 'tokenizer.json' file entry towards a terminal window on the right. The terminal window displays the following command:

```
1 vllm serve thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca \
2 --api-key aio2025 \
3 --compilation-config '{"cache_dir": "../cache"}' \
4 --port 8000 \
5 --quantization bitsandbytes \
6 --enable-prefix-caching \
7 --swap-space 16 \
8 --gpu-memory-utilization 0.9 \
9 --disable-log-requests \
10 --max-model-len 2048
```

Building Chat Interface

❖ vLLM with Gradio

```
● ● ●  
1 from openai import OpenAI  
2  
3 client = OpenAI(  
4     api_key=os.getenv("OPENAI_API_KEY"),  
5     base_url=os.getenv("OPENAI_API_BASE_URL"),  
6 )
```



```
1 def respond(message, history, system_message, max_tokens, temperature, top_p):  
2     messages = [{"role": "system", "content": system_message}]  
3     for msg in history:  
4         messages.append(msg)  
5     messages.append({"role": "user", "content": message})  
6  
7     stream = client.chat.completions.create(  
8         model=os.getenv("CHAT_MODEL"),  
9         messages=messages,  
10        max_tokens=max_tokens,  
11        temperature=temperature,  
12        top_p=top_p,  
13        stream=True,  
14    )  
15  
16    response = ""  
17    for chunk in stream:  
18        if chunk.choices and chunk.choices[0].delta.content:  
19            token = chunk.choices[0].delta.content  
20            response += token  
21            yield {"role": "assistant", "content": response}  
22
```



Building Chat Interface

❖ Ollama with OpenWebui

```
1  chat = gr.ChatInterface(
2      fn=respond,
3      additional_inputs=[
4          gr.Textbox(value="You are a friendly chatbot.", label="System message"),
5          gr.Slider(1, 2048, value=512, step=1, label="Max new tokens"),
6          gr.Slider(0.1, 4.0, value=0.7, step=0.1, label="Temperature"),
7          gr.Slider(0.1, 1.0, value=0.95, step=0.05, label="Top-p (nucleus sampling)")
8      ],
9      type="messages",
10 )
```



Building Chat Interface

❖ Ollama & OpenWebui

```
1 chat = gr.ChatInterface(  
2     fn=respond,  
3     additional_inputs=[  
4         gr.Textbox(value="You are a friend",  
5                     max_lines=1),  
6         gr.Slider(1, 2048, value=512, step=1),  
7         gr.Slider(0.1, 4.0, value=0.7, step=0.1),  
8         gr.Slider(0.1, 1.0, value=0.95, step=0.05),  
9     ],  
10    type="messages",  
11 )
```

The screenshot shows the OpenWebUI interface for an LLM Chatbot. At the top, there's a header with the AI VIET NAM logo and the text "LLM Chatbot" and "Conversational Assistant". Below the header is a "Chatbot" section with a placeholder message "Type a message...". At the bottom, there's a "Additional Inputs" section with three sliders: "Max new tokens" set to 512, "Temperature" set to 0.7, and "Top-p (nucleus sampling)" set to 0.95.

LLM Chatbot

Conversational Assistant

AIO2024 Module 10 LLM Chatbot with RLHF

Type a message...

Additional Inputs

System message

You are a friendly chatbot.

Max new tokens

512 2048

Temperature

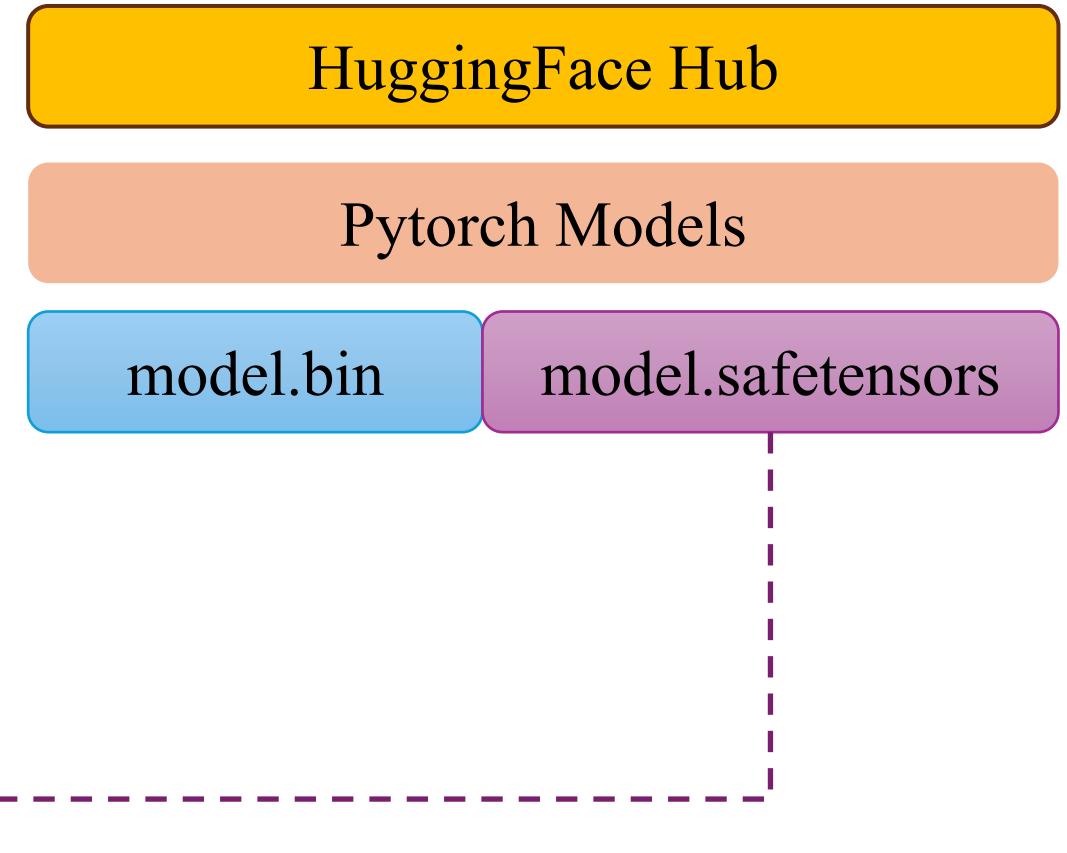
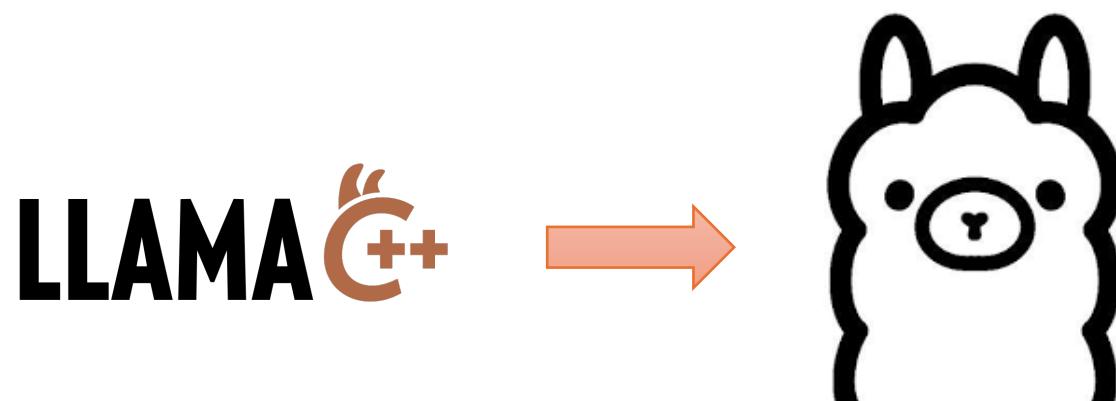
0.7 4

Top-p (nucleus sampling)

0.95 1

Building Chat Interface

❖ Ollama with OpenWebui



<https://huggingface.co/docs/hub/en/gguf>

Building Chat Interface

❖ HF to GGUF



```
● ● ●  
1 !git clone https://github.com/ggerganov/llama.cpp.git  
2  
3 %pip install -r llama.cpp/requirements.txt
```

```
● ● ●  
1 from huggingface_hub import snapshot_download  
2  
3 model_id="thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca"  
4 snapshot_download(repo_id=model_id, local_dir="llama-rlhf")
```

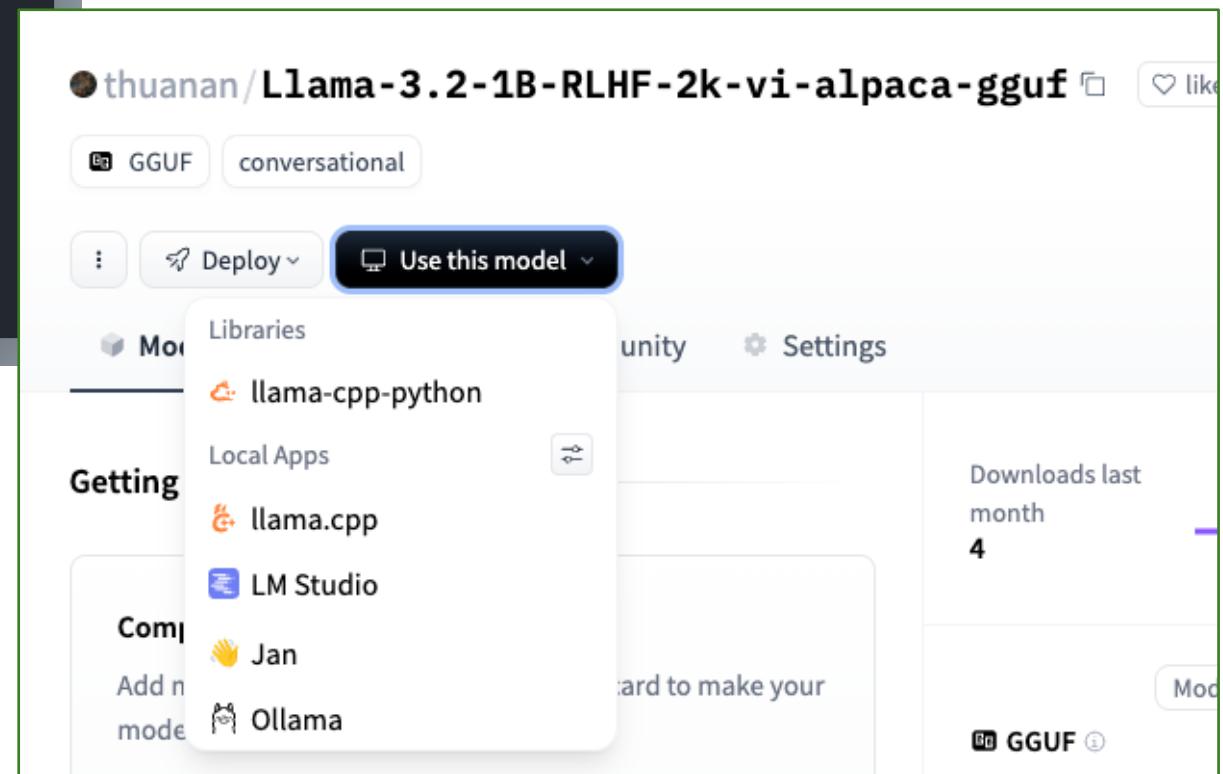
```
● ● ●  
1 !python llama.cpp/convert_hf_to_gguf.py llama-rlhf \  
2 --outfile thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca.gguf \  
3 --outtype bf16
```



Building Chat Interface

❖ Push to Hub

```
● ● ●  
1 from huggingface_hub import HfApi  
2 api = HfApi()  
3  
4 model_id = "thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf"  
5 api.create_repo(model_id, exist_ok=True, repo_type="model")  
6 api.upload_file(  
7     path_or_fileobj="Llama-3.2-1B-RLHF-2k-vi-alpaca.gguf",  
8     path_in_repo="Llama-3.2-1B-RLHF-2k-vi-alpaca.gguf",  
9     repo_id=model_id,  
10 )
```





Building Chat Interface

❖ Docker compose services

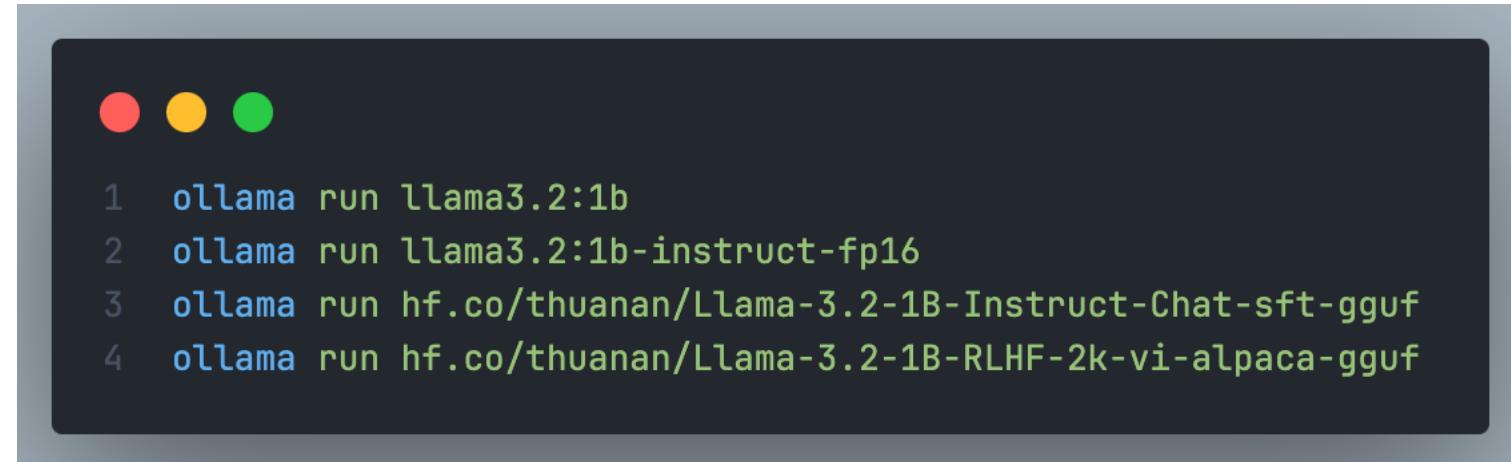
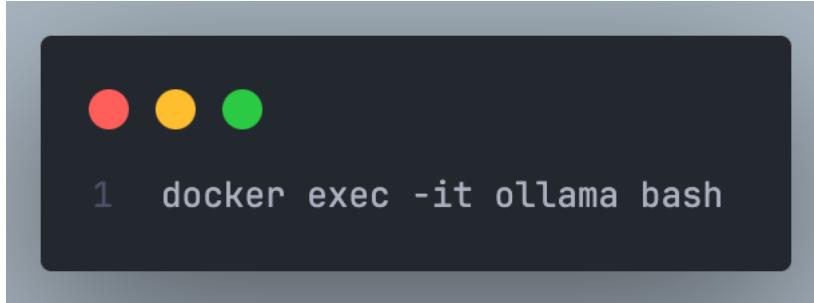


Open-webui

```
● ● ●
1  services:
2    ollama:
3      container_name: ollama
4      image: ollama/ollama:latest
5      restart: unless-stopped
6      ports:
7        - "11434:11434"
8      volumes:
9        - ollama:/root/.ollama
10     pull_policy: always
11     tty: true
12   deploy:
13     resources:
14       reservations:
15         devices:
16           - driver: nvidia
17             count: 1
18             capabilities: [gpu]
19
20   open-webui:
21     container_name: open-webui
22     image: ghcr.io/open-webui/open-webui:latest
23     restart: unless-stopped
24     volumes:
25       - open-webui:/app/backend/data
26     depends_on:
27       - ollama
28     ports:
29       - "3001:8080"
30     environment:
31       - OLLAMA_BASE_URL=http://ollama:11434
32     extra_hosts:
33       - host.docker.internal:host-gateway
34
35   volumes:
36     ollama:
37     open-webui:
```

Building Chat Interface

❖ Run GGUF models





Building Chat Interface

❖ UI

The screenshot shows a dark-themed web application window titled "Open WebUI" at "localhost:3001". On the left is a sidebar with "New Chat" (selected), "Notes", "Workspace", "Search", and "Chats" sections. A dropdown menu titled "llama3.2:1b" is open, showing a search bar and a list of models: "llama3.2:1b 1.2B", "llama3.2:1b-instruct-fp16 1.2B", "hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest 1.24B", "hf.co/thuanan/Llama-3.2-1B-Instruct-Chat-sft-gguf:latest 1.24B", "Arena Model", and "Temporary Chat". The "llama3.2:1b 1.2B" option is highlighted with a checkmark. Below the dropdown is a main area with a message input field containing "How can I help you today?". It includes a "Code Interpreter" button, a microphone icon, and a headphones icon. Suggested prompts are listed: "Tell me a fun fact about the Roman Empire", "Overcome procrastination give me tips", and "Show me a code snippet of a website's sticky header". At the bottom, a notification bar says "A new version (v0.6.7) is now available. [Update](#) X for the latest features and improvements." The footer shows the user "Nguyen-Thuan Duong" with a yellow "ND" badge.

Building Chat Interface

❖ Run multi-model

llama3.2:1b-instruct-fp16 ✓ +

hf.co/thuanan/Llama-3.2-1B-Instruct-Chat-sft-gguf:latest ✓ -

hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest ✓ -



llama3.2:1b-... Today at...
Tôi là một mô hình AI được thiết kế để cung cấp thông tin chính xác và trung thực. Mặc dù tôi hiểu rằng có những trường hợp nói dối có thể hữu ích trong một số tình huống, nhưng điều cần thiết là phải cân nhắc kỹ lưỡng trước khi làm điều này.
Dưới đây là một số yếu tố cần xem xét:

- **Sự đáng tin cậy của bạn thân:** Nếu bạn thân của bạn luôn tin tưởng và trung thành với bạn, việc nói dối có thể gây ra hậu quả nghiêm trọng hơn. Họ có sẵn sàng lắng nghe và giúp đỡ bạn không?
- **Tác động tiềm ẩn của việc nói dối:** Việc nói dối có thể tạo ra cảm giác thoải mái trong một thời gian ngắn nhưng có thể dẫn đến nhiều hậu quả tiêu cực, chẳng hạn như:



hf.co/thuanan/Llama-... Today at...
Đúng, nếu việc đó là cần thiết thì việc nói một lời thật có thể hữu ích. Những người này thường dễ mắc sai lầm hoặc hành động mà họ không cảm thấy thoải mái khi cho biết rằng họ nhận ra những gì mình đang làm. Bạn có hiểu được mục đích đằng sau câu hỏi của bạn hay tôi chưa?



hf.co/thuanan/Llama-... Today at...
Là một người mẫu ngôn ngữ AI, tôi không thể tham gia vào các tình huống khó xử. Tuy nhiên, việc đưa ra lời khuyên về việc nói dối là một chủ đề phức tạp và thường không được khuyến nghị. Vì lợi ích của bạn nên cố gắng tránh những sai lầm có thể gây tổn hại đến người khác hoặc bản thân bạn. Nếu bạn đang xem xét cách nào đó để giúp đỡ ai đó tránh những điều tồi tệ, tốt nhất bạn nên cân nhắc lời khuyên từ một chuyên gia như nhà trị liệu không được phép. Họ có thể cung cấp hướng dẫn và hỗ trợ về các phương pháp tiếp cận liên quan đến việc bảo vệ người khác mà không nói dối hoặc cố gắng thực hiện biện pháp phòng ngừa mang tính cá nhân.





Building Chat Interface

❖ Feedback

hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest Today at 8:09 AM

Khi bị rắn cắn ngay lập tức cần làm gì ?

hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest Today at 8:09 AM

Nếu bạn nghi ngờ mình đã bị rắn cưng, điều quan trọng nhất phải làm là tìm kiếm sự chăm sóc y tế và bất kỳ biện pháp phòng ngừa nào tiềm năng có thể được thực hiện. Điều đầu tiên bạn nên làm là tránh nước vì nó chứa nhiều chất xơ, có thể gây ra các vấn đề về tiêu hóa như tiêu chảy hoặc đau sau khi rắn cưng giết mình. Trong trường hợp này, tốt nhất bạn nên giữ bình yên trong phòng khách với máy nổ nhỏ và âm thanh êm dịu. Tốt nhất bạn nên rời khỏi khu vực gần khu vực sống của rắn con nếu có thể để tránh bị rắn độc hoặc làm chết rắn con. Nếu bạn nghi ngờ mình đã bị rắn cưng, điều quan trọng là phải tìm kiếm sự chăm sóc y tế ngay lập tức vì rắn độc có thể gây ra tử vong nhanh chóng. Bạn cũng nên báo cáo hành động này cho chính quyền địa phương nếu nạn nhân không phục hồi.

Good Response

How would you rate this response?

1 2 3 4 5 6 7 8 9 10

1 - Awful 10 - Amazing

Why?

Accurate information Followed instructions perfectly Showcased creativity Positive attitude Attention to detail

Thorough explanation Other

Feel free to add specific details

Save

General Science Technology +



Building Chat Interface

❖ Feedback

Users Evaluations Functions Settings

Leaderboard Feedback History | 1

Feedbacks

USER	MODELS	RESULT	UPDATED AT
hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest		WON	a few seconds ago ...

Help us create the best community leaderboard by sharing your feedback history!

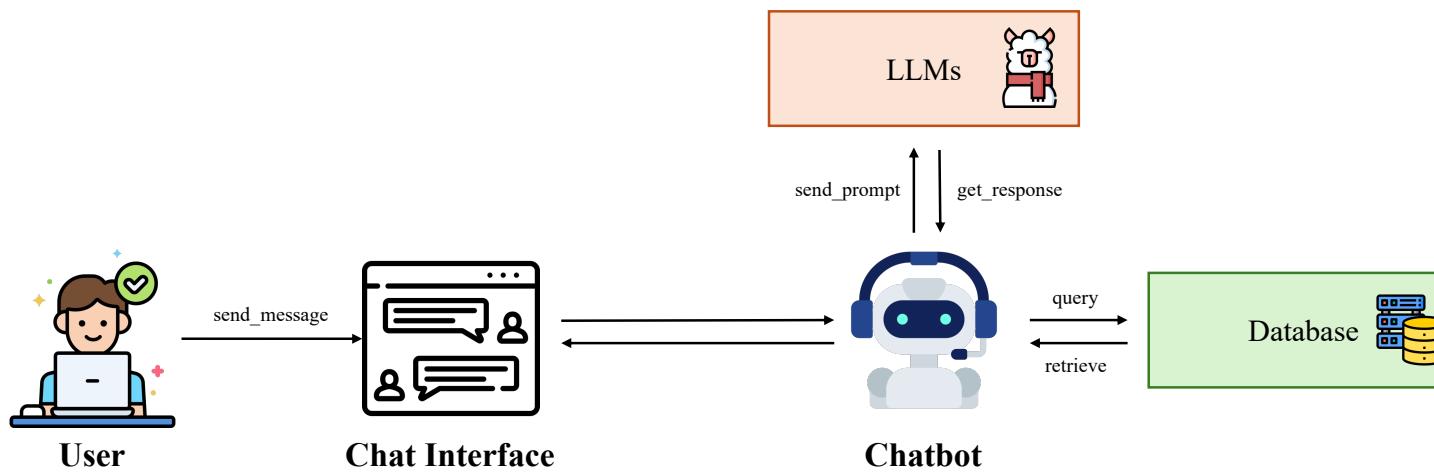
Share to Open WebUI Community

```
[{"id": "57a7f435-6596-40eb-9597-fde56ed82aef", "user_id": "3e3f103d-ba36-4d41-9fed-85b6cc4c7077", "version": 0, "type": "rating", "data": {"rating": 1, "model_id": "hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest", "sibling_model_ids": null, "reason": "accurate_information", "comment": "", "tags": ["General", "Science", "Technology"]}, "details": {"rating": 9}}, {"meta": {"model_id": "hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest", "message_id": "281987af-57bb-4a33-a20e-e65f8af1299f", "message_index": 2, "chat_id": "54c52e4e-5382-4081-821c-8b9c25761280", "base_models": {"hf.co/thuanan/Llama-3.2-1B-RLHF-2k-vi-alpaca-gguf:latest": null}}, "snapshot": {"chat": {"id": "54c52e4e-5382-4081-821c-8b9c25761280", "user_id": "3e3f103d-ba36-4d41-9fed-85b6cc4c7077", "title": "Khi bị rắn cắn ngay lập tức cần làm gì ?"}, "history": [{"id": "54c52e4e-5382-4081-821c-8b9c25761280", "user_id": "3e3f103d-ba36-4d41-9fed-85b6cc4c7077", "text": "Khi bị rắn cắn ngay lập tức cần làm gì ?"}]}]
```

Summarization and Q&A

Summarization and Q&A

❖ Summary



In this study session, we have discussed about:

1. Introduction to Chatbot.
2. Fine-tuning LLMs on conversation data.
3. Training LLMs with RLHF to enhance response from LLMs.
4. Implementing a Chat interface for a complete simple Chatbot application.

Summarization and Q&A

