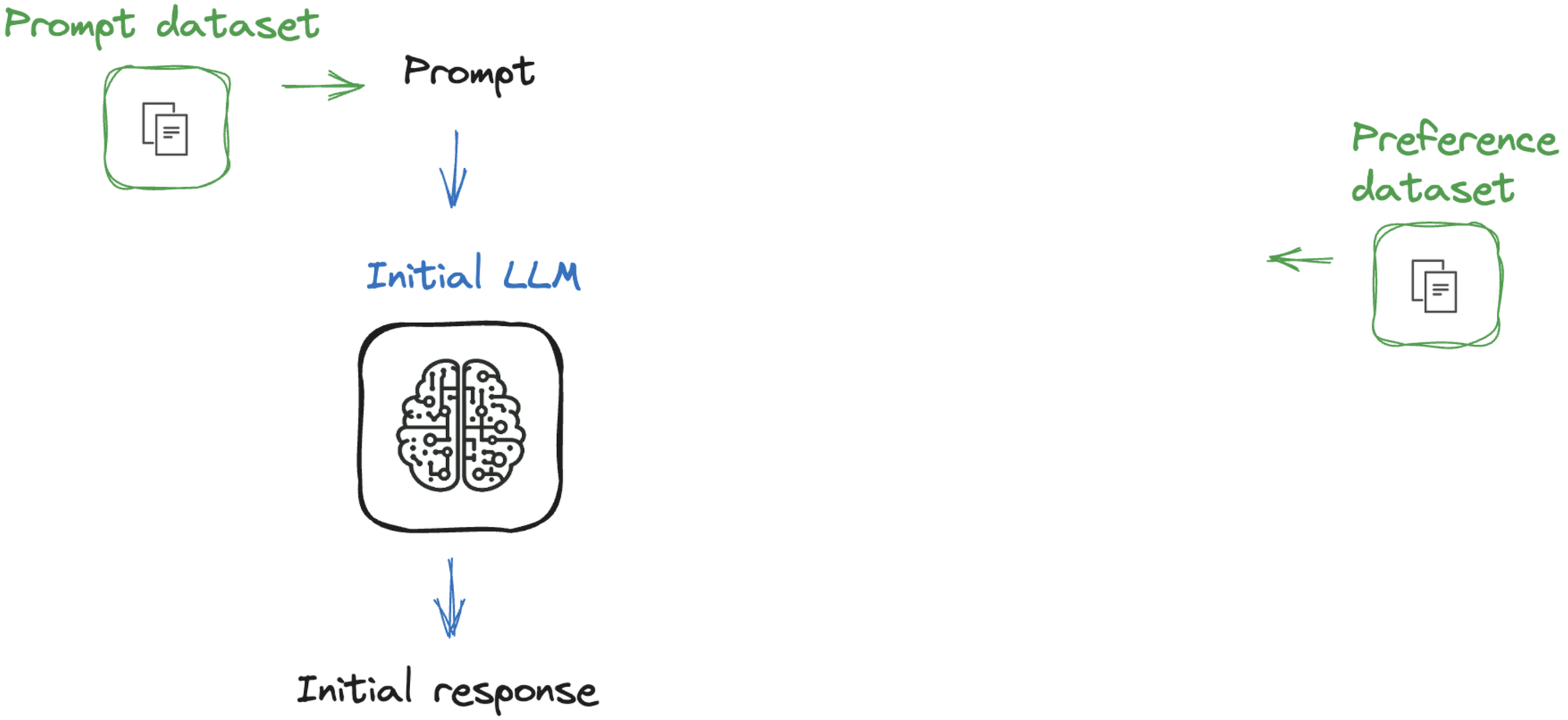
Reward models explored

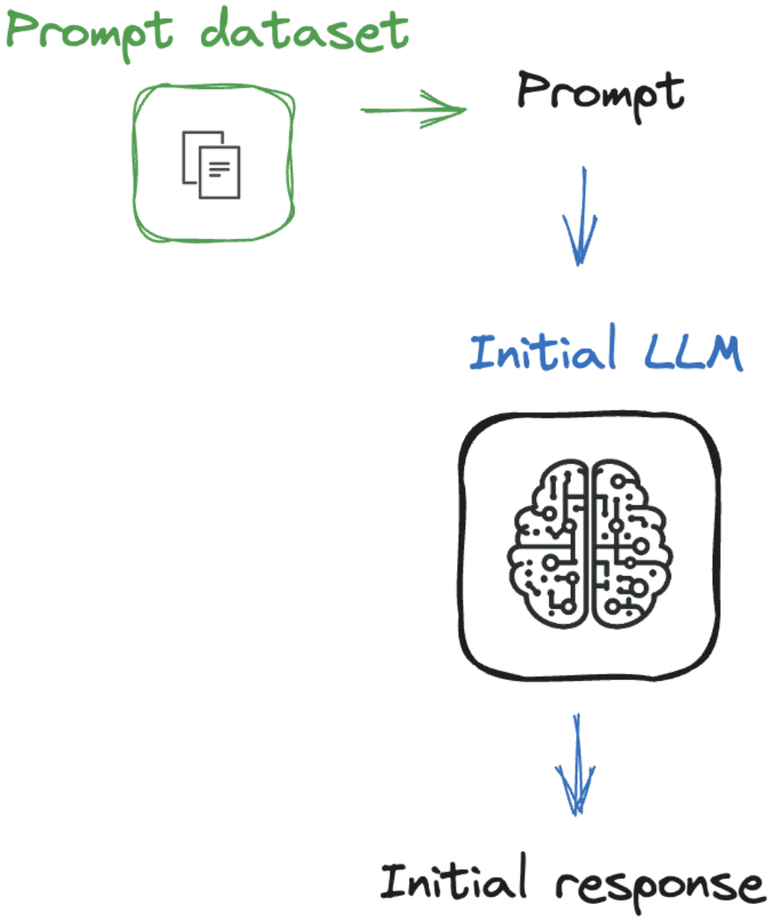
REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

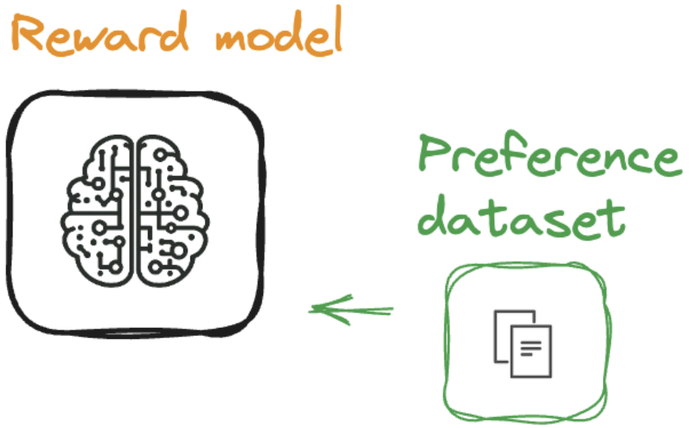
Mina Parham
AI Engineer

Process so far

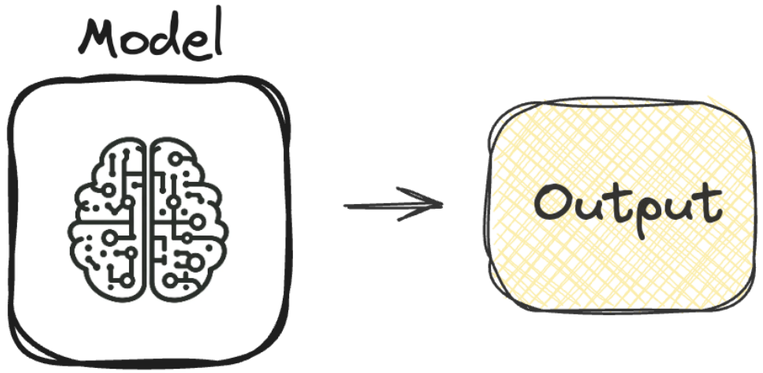


Process so far



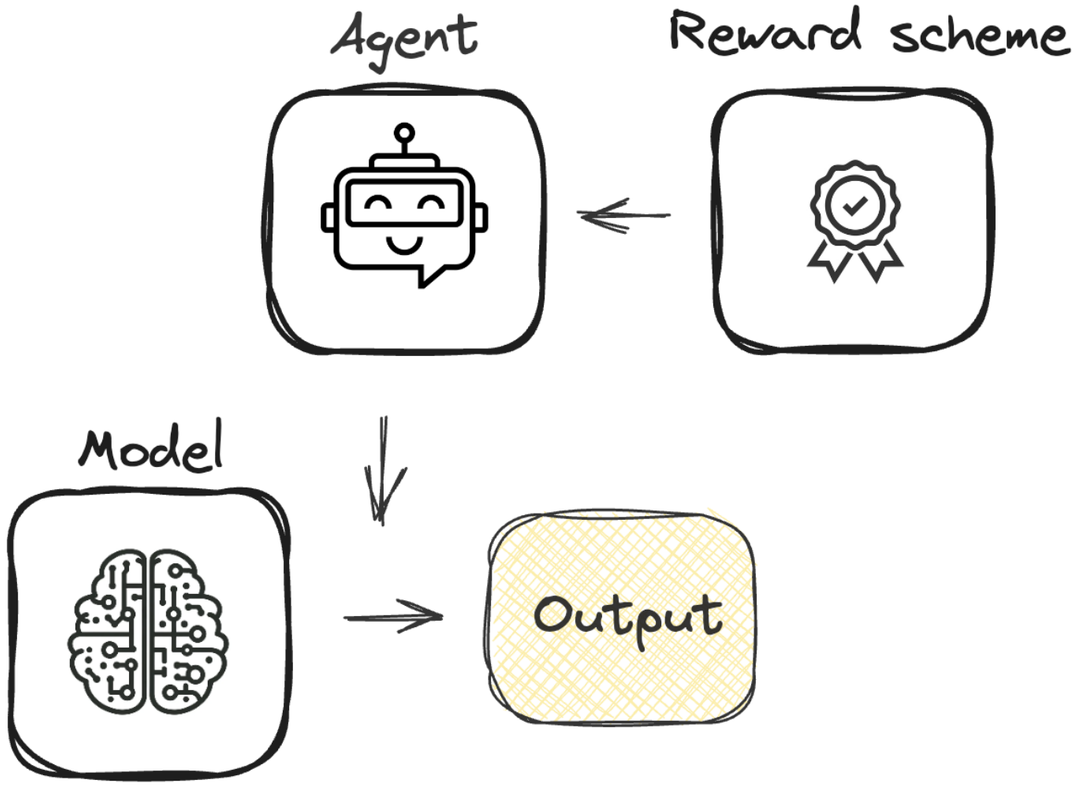


What is a reward model?



What is a reward model?

- **Model** informs the **agent**
- **Agent** evaluates the **model** to maximize **rewards**



Using the reward trainer

```
from trl import RewardTrainer, RewardConfig
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from datasets import load_dataset
```

```
# Load pre-trained model and tokenizer
model = AutoModelForSequenceClassification.from_pretrained("gpt2", num_labels=1)
tokenizer = AutoTokenizer.from_pretrained("gpt2")
# Load dataset in the required format
dataset = load_dataset("path/to/dataset")
```

Training the reward model

```
# Define training arguments
training_args = RewardConfig(
    output_dir="path/to/output/dir",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    learning_rate=1e-3
)
```

Training the reward model

```
# Initialize the RewardTrainer
trainer = RewardTrainer(
    model=model,
    args=training_args,
    train_dataset=dataset["train"],
    eval_dataset=dataset["validation"],
    tokenizer=tokenizer,
)
```

```
# Train the reward model
trainer.train()
```

Let's practice!

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

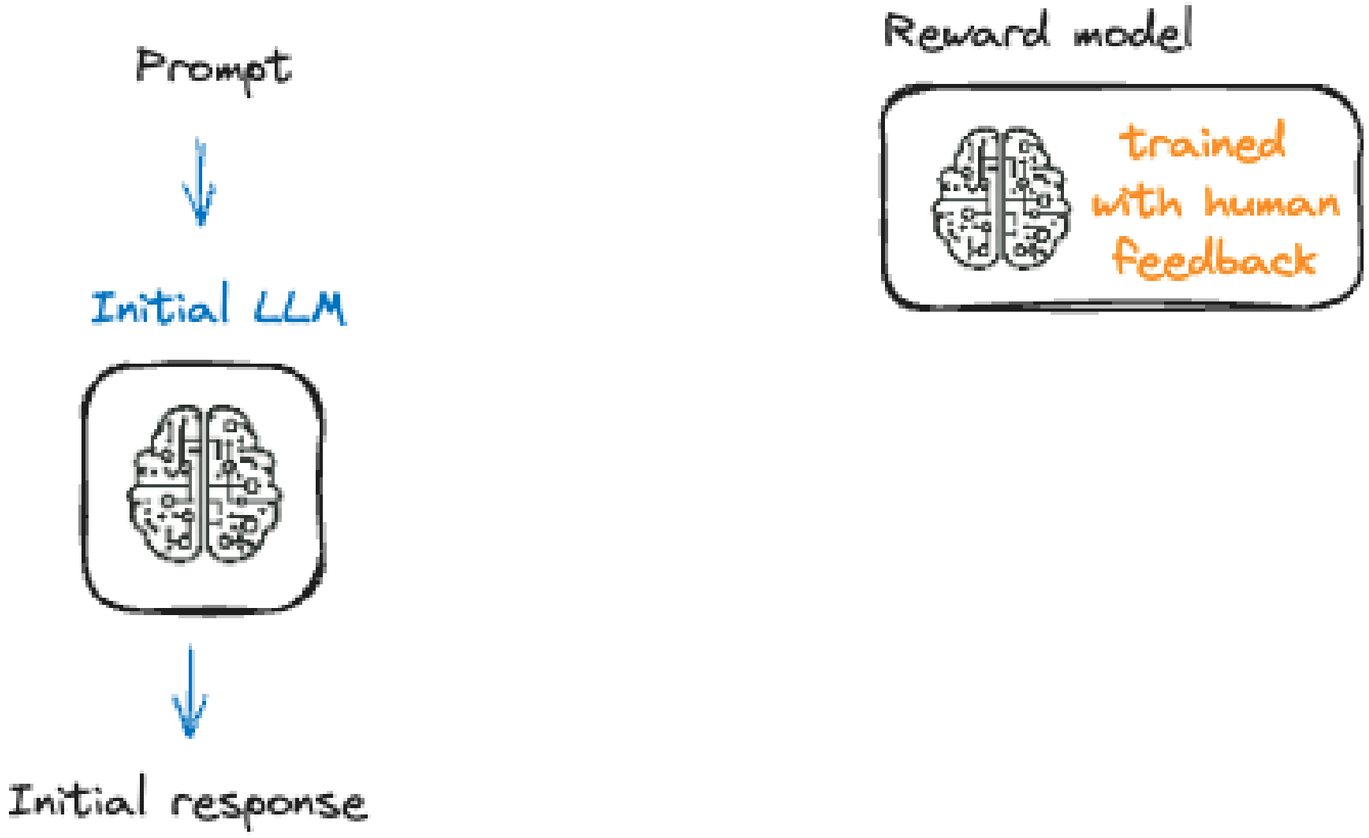
Training with PPO

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

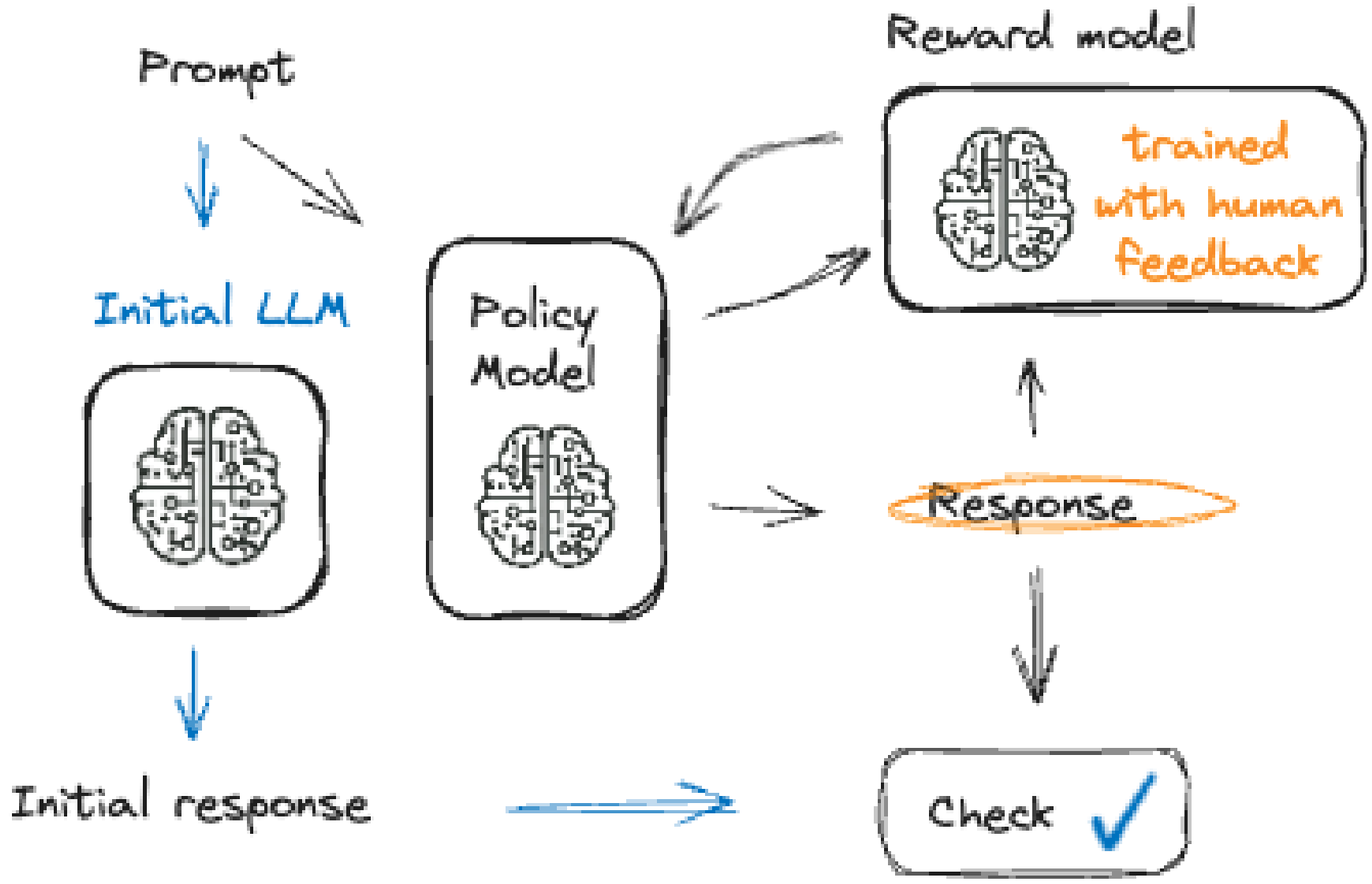
Mina Parham
AI Engineer

datacamp

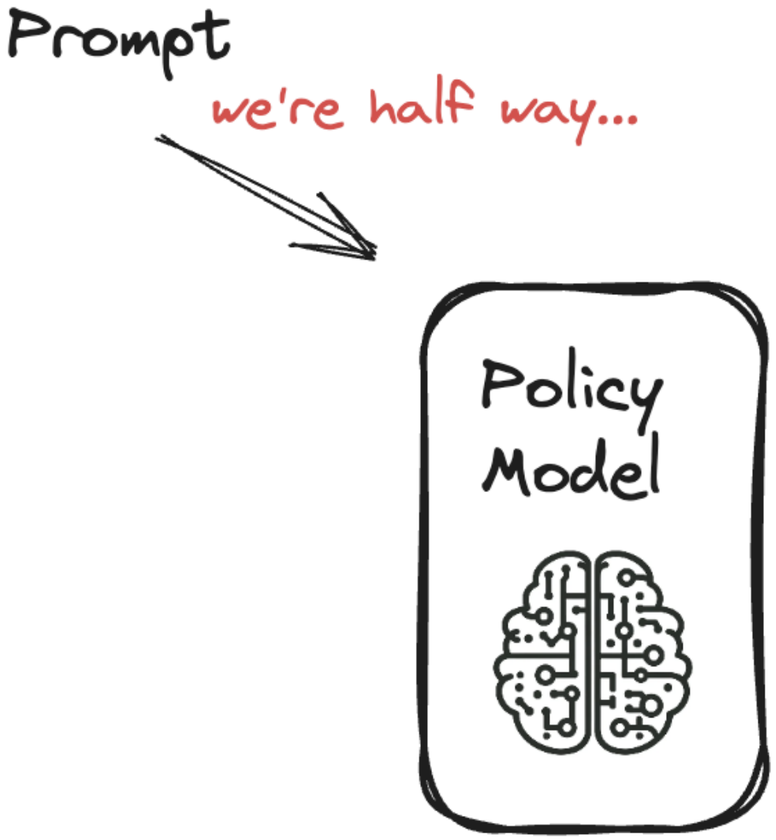
Fine-Tuning with reinforcement learning



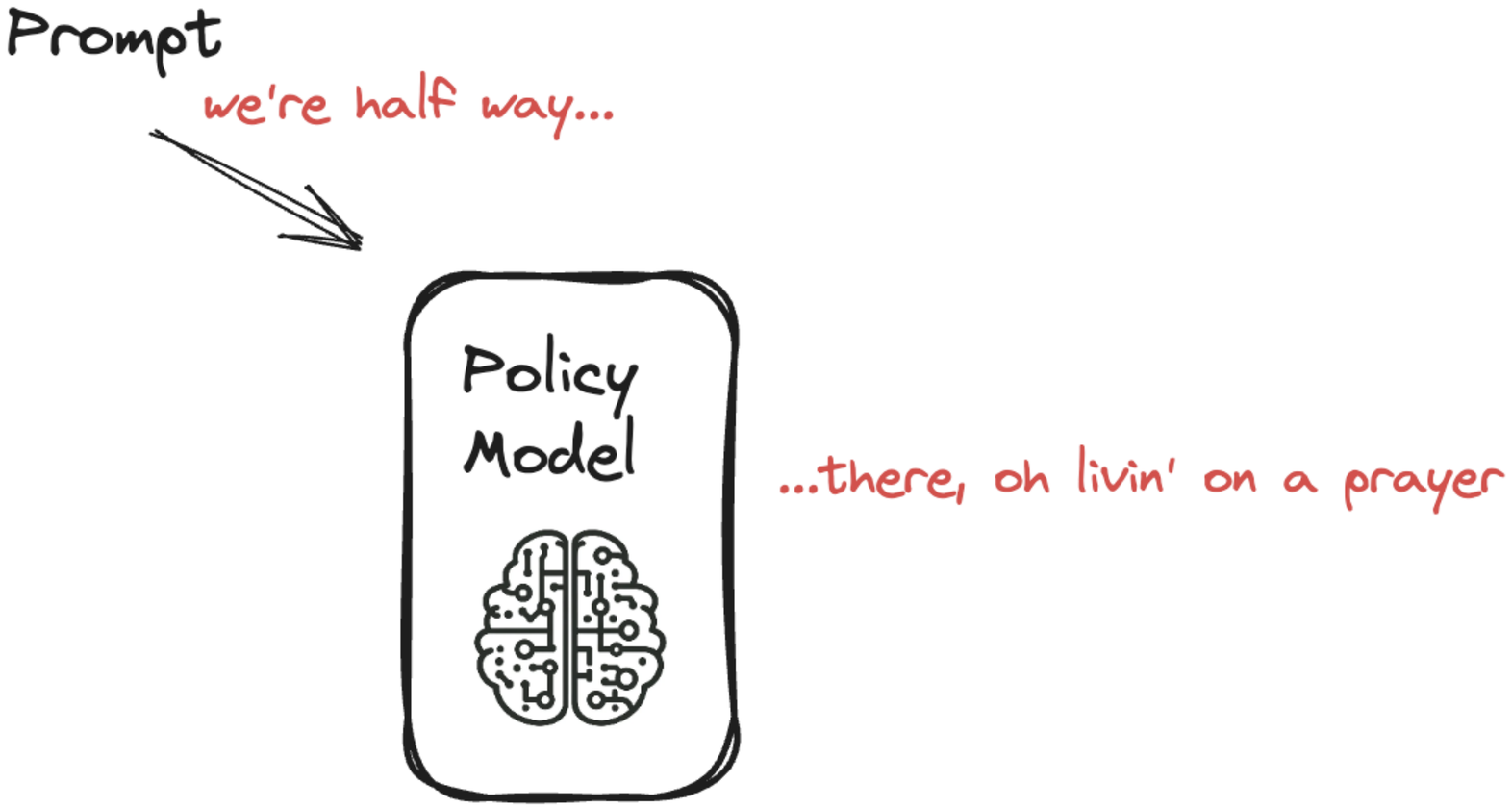
Fine-Tuning with reinforcement learning



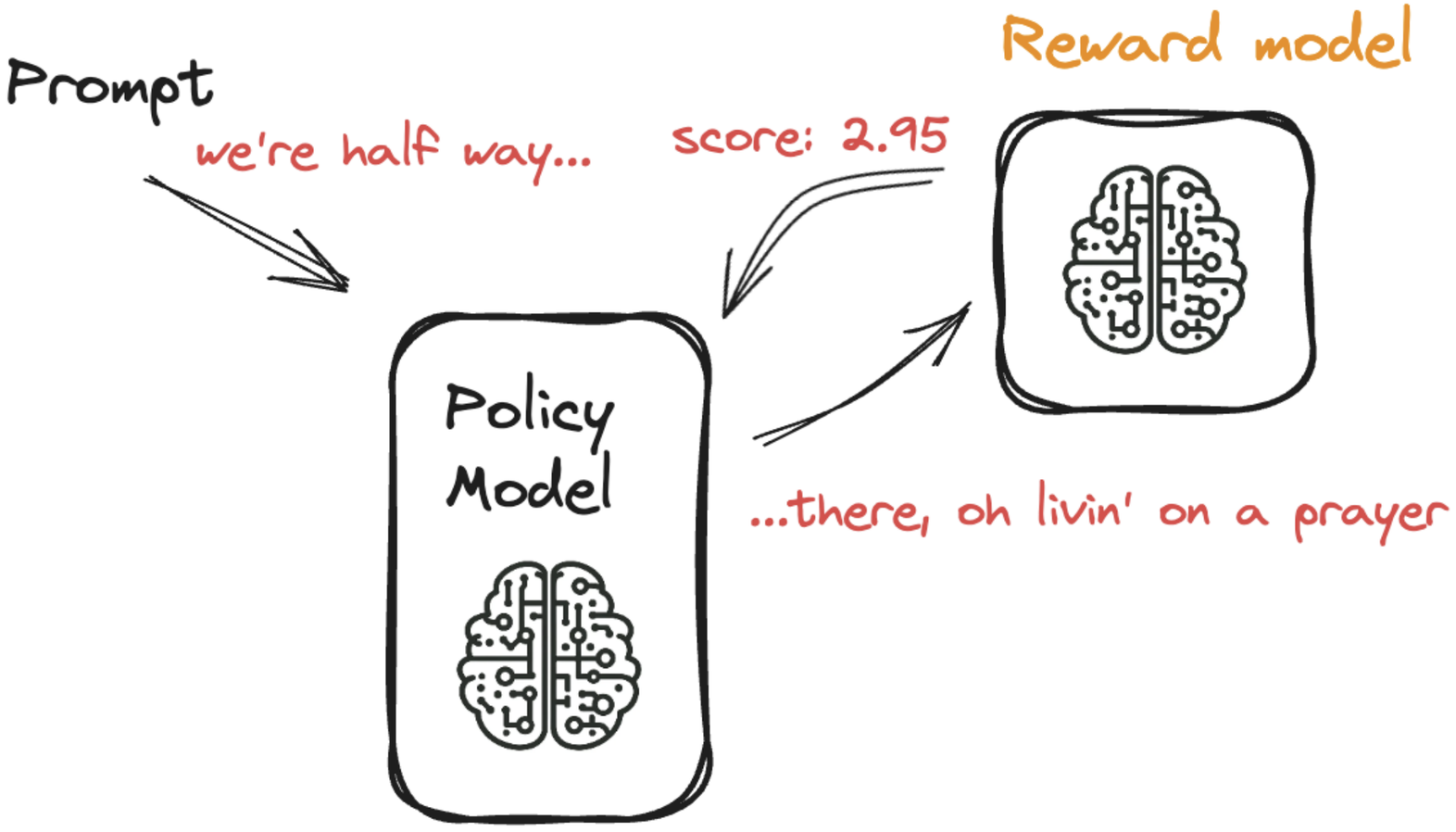
Fine-Tuning a Language Model with PPO



Fine-Tuning a Language Model with PPO



Fine-Tuning a Language Model with PPO



Fine-Tuning a Language Model with PPO

- PPO: gradual adjustment for the model
- Avoids overfitting to feedback

Implementing PPOTrainer with TRL

```
from trl import PPOConfig
config = PPOConfig(model_name="gpt2",learning_rate=1.4e-5)
```

```
from trl import AutoModelForCausalLMWithValueHead
model = AutoModelForCausalLMWithValueHead.from_pretrained(config.model_name)
tokenizer = AutoTokenizer.from_pretrained(config.model_name)
```

```
from trl import PPOTrainer
ppo_trainer = PPOTrainer(model=model,config=config,dataset=dataset,
                         tokenizer=tokenizer)
```

Starting the training loop

```
for epoch in tqdm(range(10), "epoch: "):

  for batch in tqdm(ppo_trainer.dataloader):
    # Get responses
    response_tensors = ppo_trainer.generate(batch["input_ids"])
    batch["response"] = [tokenizer.decode(r.squeeze()) for r in response_tensors]
    # Compute reward score
    texts = [q + r for q, r in zip(batch["query"], batch["response"])]
    rewards = reward_model(texts)
    stats = ppo_trainer.step(query_tensors, response_tensors, rewards)
    ppo_trainer.log_stats(stats, batch, rewards)
```

Let's practice!

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

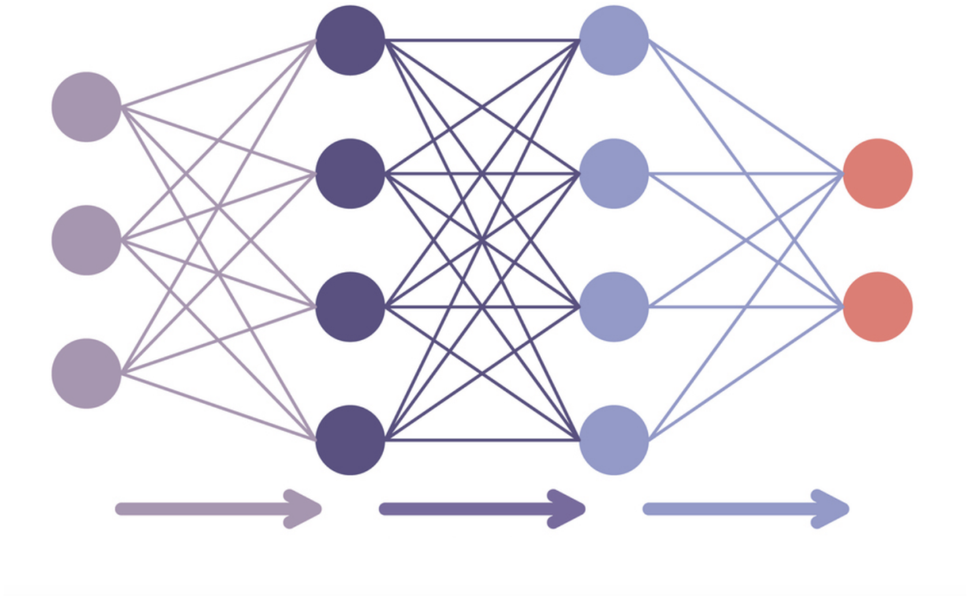
Efficient fine-tuning in RLHF

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

Mina Parham
AI Engineer

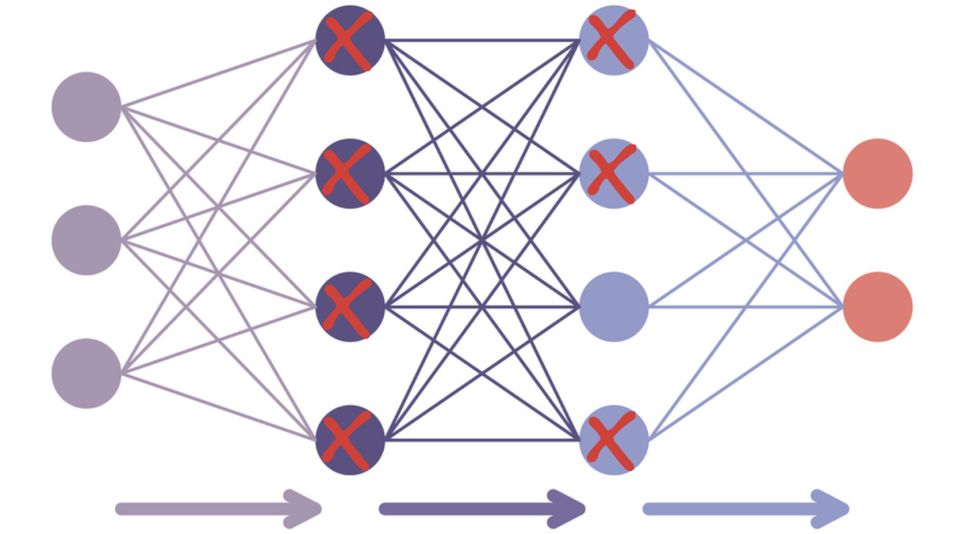
Parameter-efficient fine-tuning

- Fine-tuning a full model



Parameter-efficient fine-tuning

- Fine-tuning with PEFT



- LoRA: adjusts only a few layers
- Quantization: lowers data type precision

Step1: load your active model in 8-bit precision

```
from peft import prepare_model_for_int8_training

pretrained_model = AutoModelForCausalLM.from_pretrained(
                                                        model_name,
                                                        load_in_8bit=True
                                                       )
pretrained_model_8bit = prepare_model_for_int8_training(pretrained_model)
```

Step 2: add extra trainable adapters using peft

```
from peft import LoraConfig, get_peft_model

config = LoraConfig(
    r=32, # Rank of the low-rank matrices
    lora_alpha=32, # Scaling factor for the LoRA updates
    lora_dropout=0.1, # Dropout rate for LoRA layers
    bias="lora_only"# Only update bias terms for LoRA layers, others remain frozen
)

lora_model = get_peft_model(pretrained_model_8bit, config)
model = AutoModelForCausalLMWithValueHead.from_pretrained(lora_model)
```

Step 3: use one model for reference and active logits

```
ppo_trainer = PPOTrainer(
    config, # The config we just defined
    model, # Our PPO model
    ref_model=None,
    tokenizer=tokenizer,
    dataset=dataset,
    data_collator=collator,
    optimizer=optimizer
)
```

Let's practice!

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)