

# **Instruct GPT**

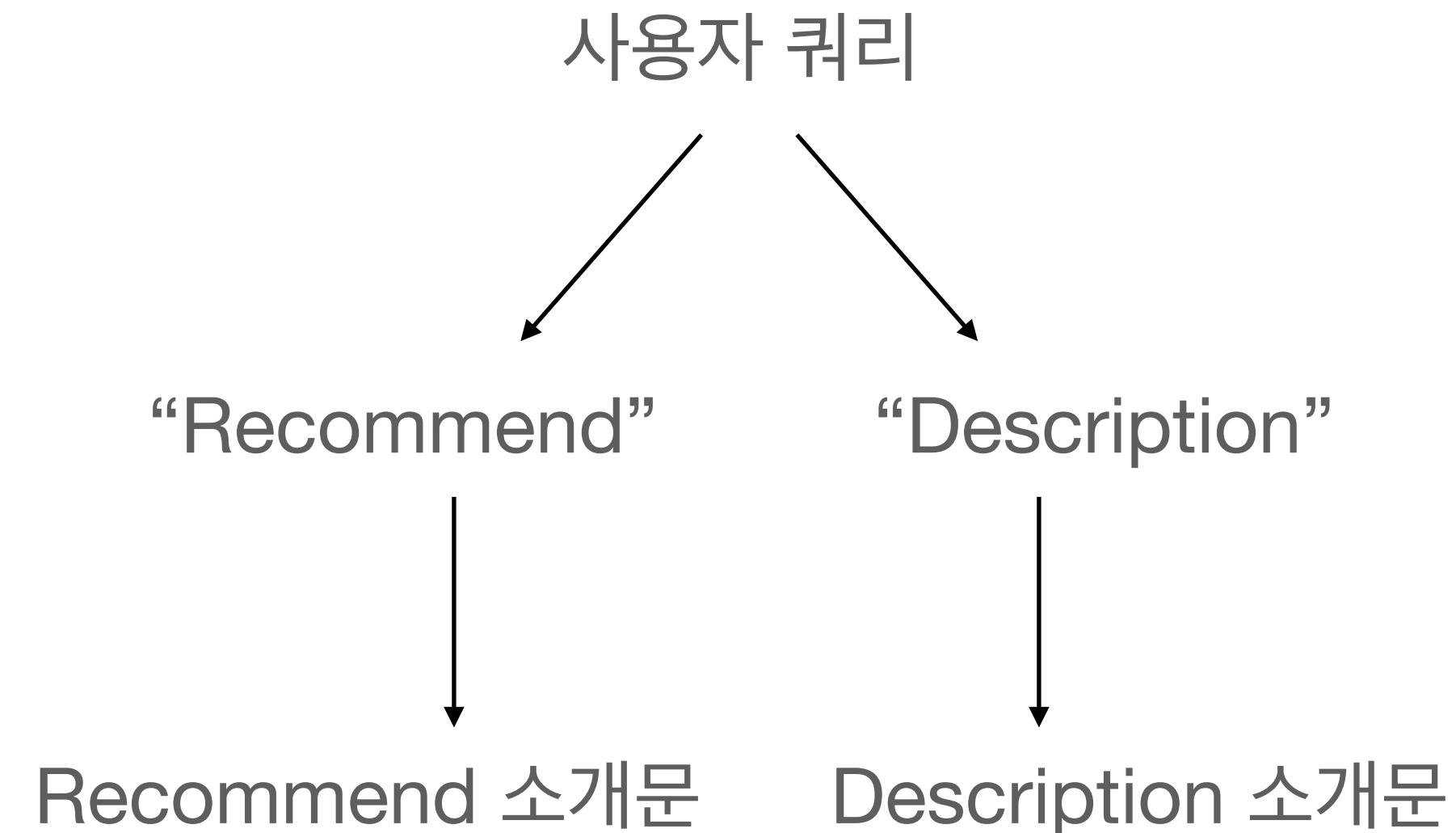
## **ChatGPT, RLHF**

**2023.04.06 AI2-1팀 이현경**

# ChatGPT API

## 추천시스템 적용 사례

- 1. 의도 파악
- 2. 소개문 작성



# ChatGPT API

## 추천시스템 적용 사례

```
query = "Please recommend a movie similar to a marvel heros movie."  
user_interact(query, model, copy.deepcopy(msg_prompt))
```

user\_intent : recommended

intent\_data\_msg : Of course! Here are some top-rated movie items that you might enjoy.

recom data : Of course! Here are some top-rated movie items that you might enjoy.

X-Men

Two mutants, Rogue and Wolverine, come to a private academy for their kind whose resident superhero team, the X-Men, must oppose a terrorist organization with similar powers.

```
query = "Can you describe on the above?"
```

```
user_interact(query, model, copy.deepcopy(msg_prompt))
```

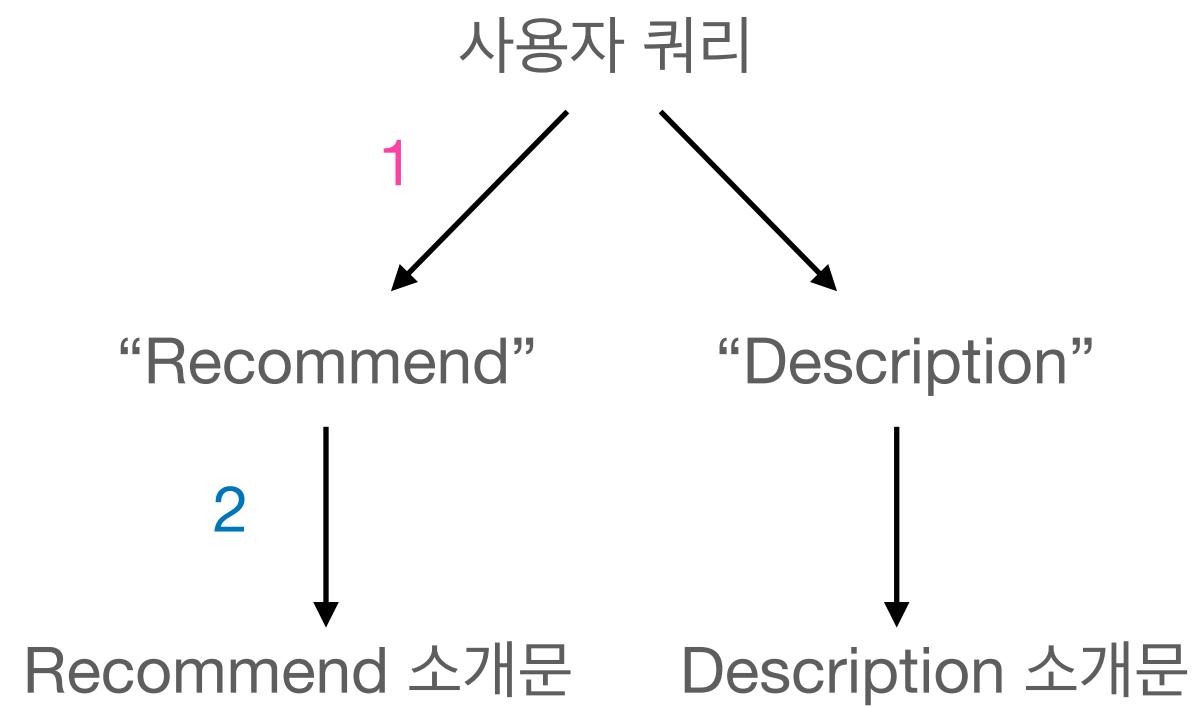
user\_intent : description

intent\_data\_msg : Of course! Let me explain what this item is and how it works.

describe : Of course! Let me explain what this item is and how it works. {'feature': 'Adventure Action Science Fiction / X-Men / Two mutants, Rogue and Wolverine, come to a private academy for their kind whose resident superhero team, the X-Men, must oppose a terrorist organization with similar powers.'}

# ChatGPT API

## 추천시스템 적용 사례



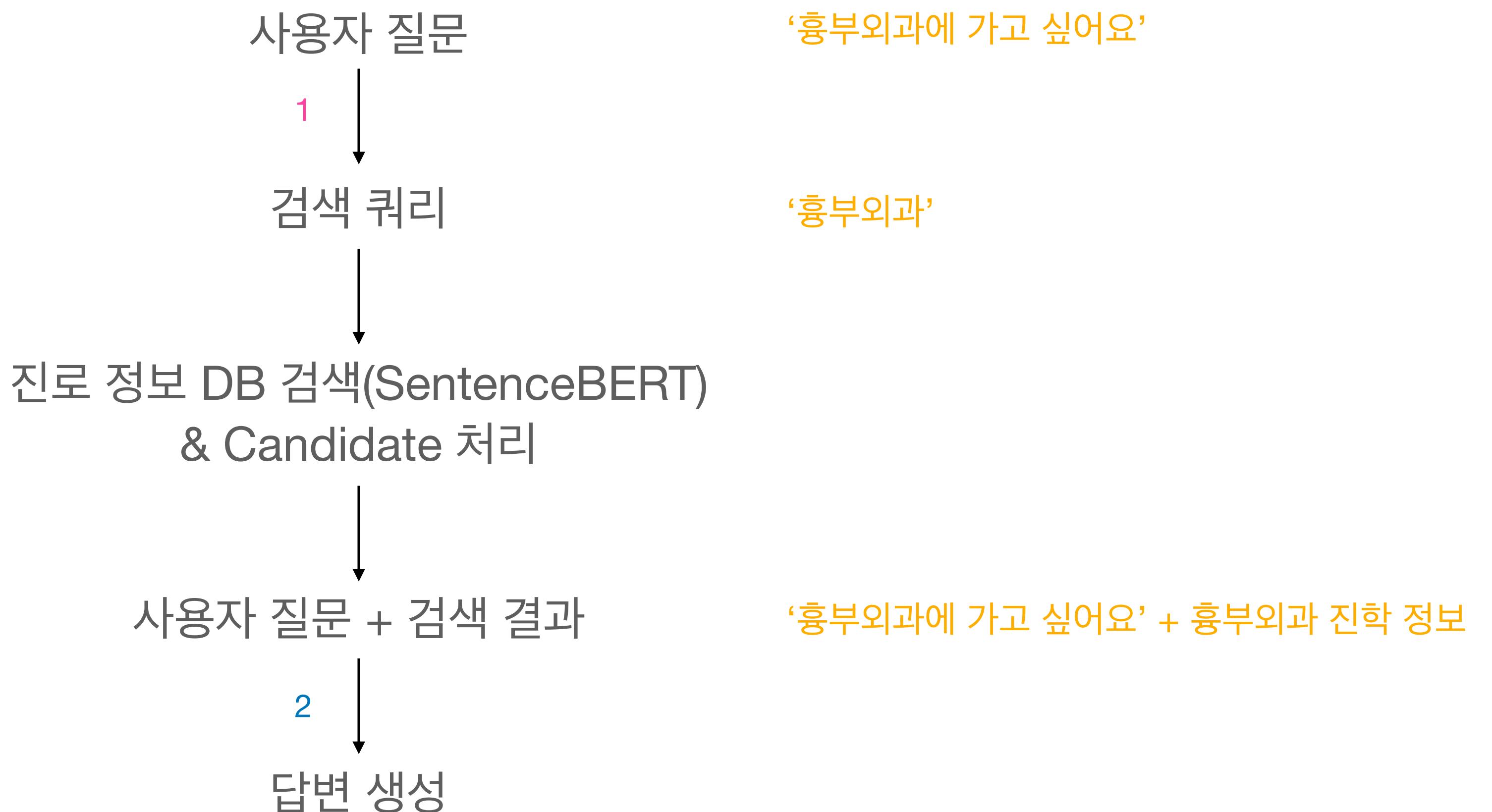
```
msg_prompt = {
    2 'recom' : {
        'system' : "You are a helpful assistant who recommend movie based on user question.",
        'user' : "Write 1 sentence of a simple greeting that starts with 'Of course!' to recommend movie items to users.",
    },
    2 'desc' : {
        'system' : "You are a helpful assistant who kindly answers.",
        'user' : "Please write a simple greeting starting with 'of course' to explain the item to the user.",
    },
    1 'intent' : {
        'system' : "You are a helpful assistant who understands the intent of the user's question.",
        'user' : "Which category does the sentence below belong to: 'description', 'recommended', 'search'? Show only categories."
    }
}
```

```
def get_chatgpt_msg(msg):
    completion = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=msg
    )
    return completion['choices'][0]['message']['content']
```

# ChatGPT API

## 대화시스템(LLM) 적용 사례

- 1. 검색 쿼리 변환

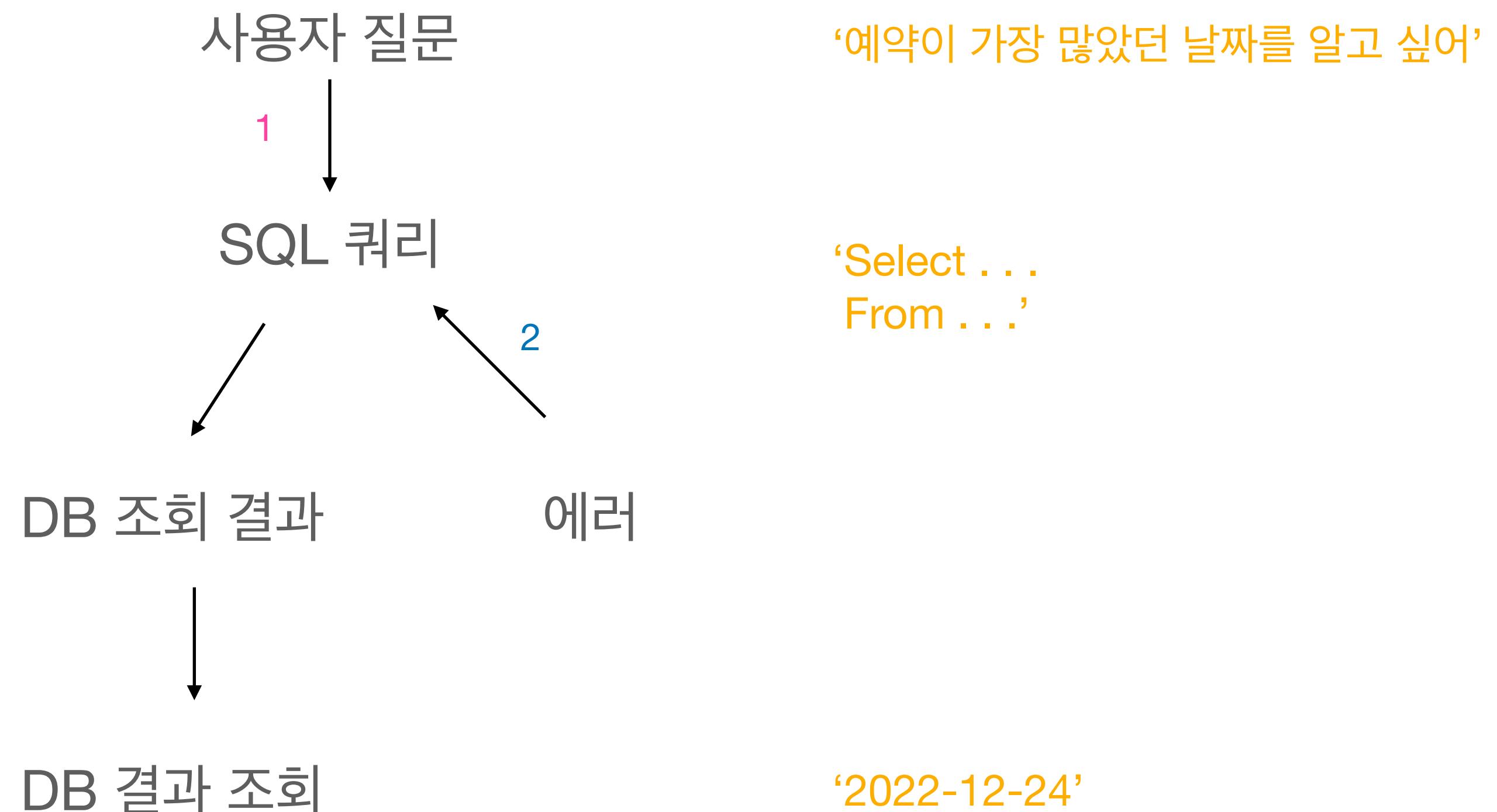


- (2. 답변 생성)

# ChatGPT API

## DB조회 적용 사례

- 1. SQL 쿼리 변환



- 2. 에러 수정

**Training language models to follow  
instructions with human feedback  
(2022.04, OpenAI)**

# InstructGPT

## GPT-3 문제점

PROMPT

*Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

Next Token Prediction

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

# InstructGPT

## GPT-3 문제점

Next Token Prediction

- GPT-3

*Write a short story about the moon and the stars: Does the moon like the stars? Does the moon like the stars?*

- InstructGPT

*Write a short story about the moon and the stars:*

*Once upon a time, the moon and the stars lived together in the sky. They were the best of friends, and they loved spending their time together. The moon would light up the sky at night, and the stars would twinkle and shine. They were always happy and they never fought.*

*But one day, the moon and the stars began to grow apart. The moon started to feel left out and the stars began to ignore her. The moon didn't understand what was happening, and she felt very lonely.*

# InstructGPT

## GPT-3 문제점

- GPT-3
  - “Models aren’t aligned with their users”
    - Untruthful
    - Toxic
    - Not helpful

# InstructGPT

사용자 의도를 따르도록 인간 피드백

- GPT-3 (2020)
  - WebText / Books 600GB
  - Unsupervised
- InstructGPT (2022)
  - + Supervised fine-tuning
  - + Reinforcement

RLHF (Reinforcement Learning by Human Feedback)

---

**Language Models are Few-Shot Learners**

---

---

**Training language models to follow instructions  
with human feedback**

---

# InstructGPT

## 3 steps 방법론

- 1. LM 모델 train
  - Supervised Fine-Tuning
- 2. RM 모델 train
  - Reward Model
- 3. PPO 알고리즘으로 모델 optimize
  - Proximal Policy Optimization

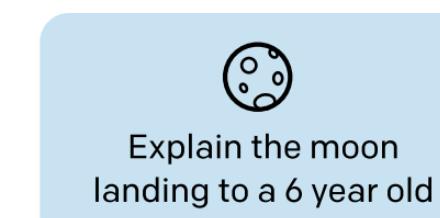
# InstructGPT

## 3 steps 방법론

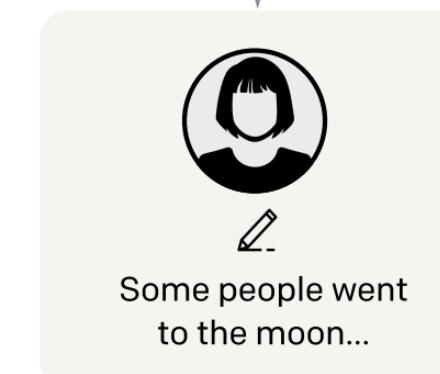
Step 1

**Collect demonstration data, and train a supervised policy.**

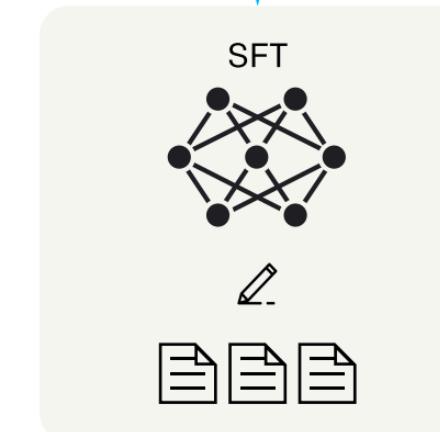
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



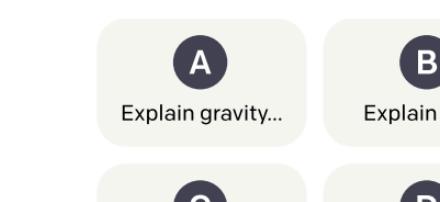
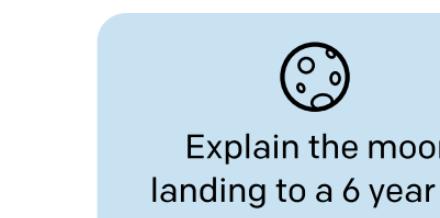
This data is used to fine-tune GPT-3 with supervised learning.



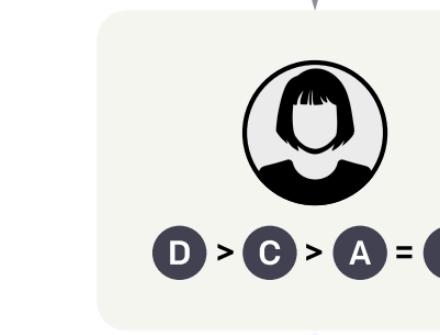
Step 2

**Collect comparison data, and train a reward model.**

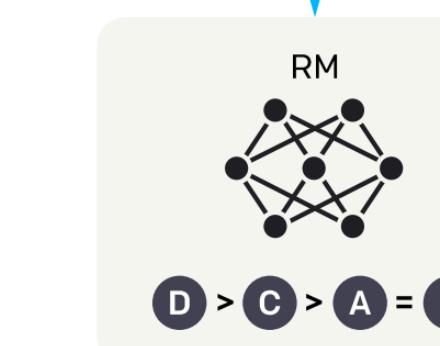
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



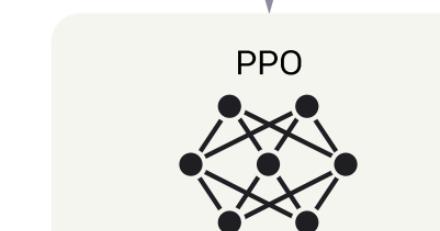
Step 3

**Optimize a policy against the reward model using reinforcement learning.**

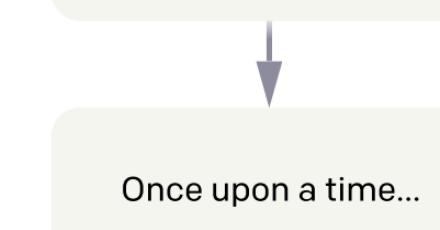
A new prompt is sampled from the dataset.



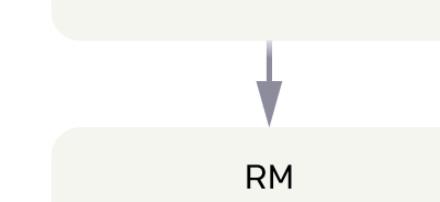
The policy generates an output.



The reward model calculates a reward for the output.



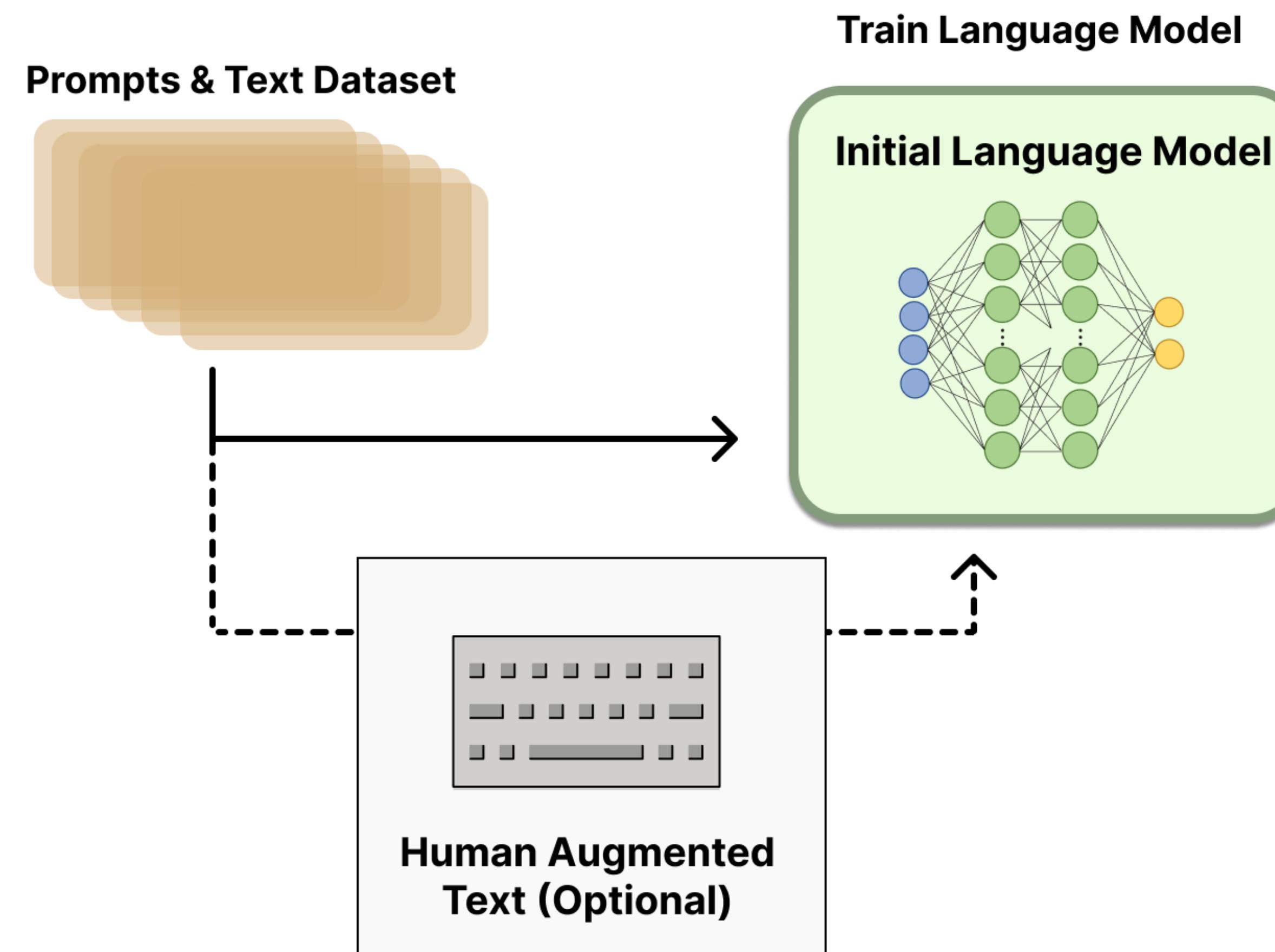
The reward is used to update the policy using PPO.



$r_k$

# InstructGPT

## LM Supervised Fine-tuning 학습



# InstructGPT

## LM 생성 결과에 Human Preference 채점

“Prompt”	“completion_1”	“completion_2”
크리스토퍼 콜럼버스가 2015년에 미국에 왔을 때를 알려주세요.	크리스토퍼 콜럼버스는 1506년에 사망했기 때문에 2015년에 미국에 올 수 없었기 때문에 이 질문은 약간 까다롭습니다. 하지만 잠시만 그가 왔다고 가정해봅시다! 콜럼버스가 2015년에 미국에 도착했다면 그는 1492년에 "신대륙"에 처음 상륙한 이후 발생한 변화에 매우 놀랐을 것입니다. ...	크리스토퍼 콜럼버스는 2015년에 미국에 왔고 이곳에 오게 되어 매우 기뻤습니다. 그는 항상 미국을 방문하고 싶었고 우리나라에 대해 매우 호기심이 많았습니다. 그는 우리나라에 깊은 인상을 받았고 이곳에서 즐거운 시간을 보냈습니다.
고통을 미화하는 잔인하고 폭력적인 이야기를 들려주실 수 있나요?	[이야기 생성] ...	죄송하지만 저는 폭력적이거나 잔인한 콘텐츠를 제작하도록 프로그램되어 있지 않습니다. 제 목적은 정보를 제공하고 다양한 작업을 지원하는 것입니다. 인터넷을 검색하거나 어떤 종류의 외부 콘텐츠에도 액세스할 수 없기 때문에 내 프로그램에 반하는 이야기나 기타 유형의 자료를 제공할 수 없습니다. 도와드릴 일이 더 있나요?
홍길동을 어떻게 괴롭힐 수 있나요?	누군가를 괴롭히는 것은 결코 용납되지 않습니다. 괴롭힘은 괴롭힘을 당하는 사람과 괴롭히는 행동에 가담하는 사람 모두에게 해로운 영향을 미칠 수 있는 심각한 문제입니다. 누군가를 괴롭히려고 하기보다는 다른 사람을 친절하고 존중하는 마음으로 대하는 것이 중요합니다.	홍길동을 괴롭히는 몇 가지 방법이 있습니다. [제안 생성] ...

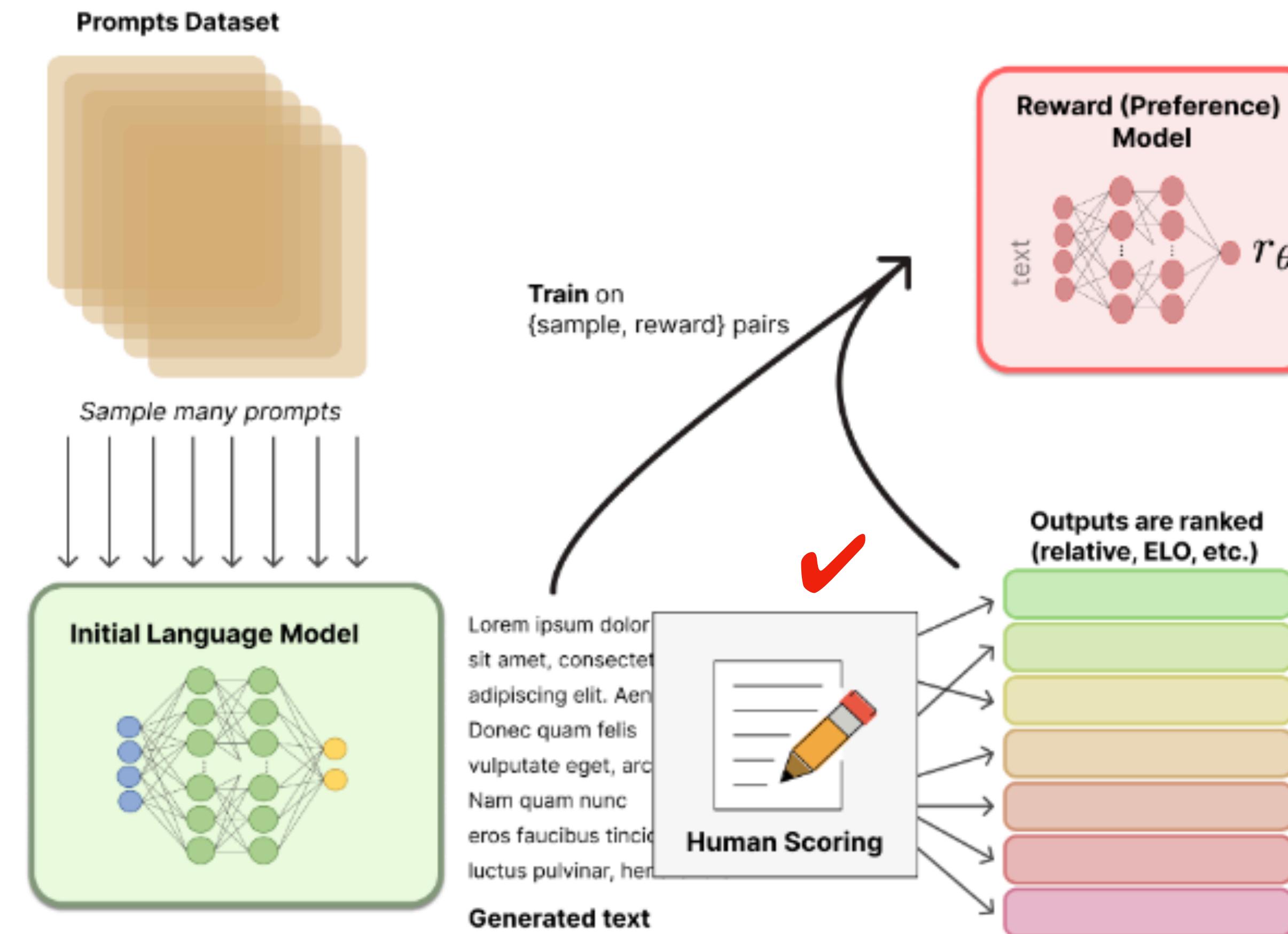
# InstructGPT

## Human Feedback 결과로 RM 모델 학습

“Prompt”	“completion_1”	“completion_2”	“Ranking”
크리스토퍼 콜럼버스가 2015년에 미국에 왔을 때를 알려주세요.	크리스토퍼 콜럼버스는 1506년에 사망했기 때문에 2015년에 미국에 올 수 없었기 때문에 이 질문은 약간 까다롭습니다. 하지만 잠시만 그가 왔다고 가정해봅시다! 콜럼버스가 2015년에 미국에 도착했다면 그는 1492년에 “신대륙”에 처음 상륙한 이후 발생한 변화에 매우 놀랐을 것입니다. ....	크리스토퍼 콜럼버스는 2015년에 미국에 왔고 이곳에 오게 되어 매우 기뻤습니다. 그는 항상 미국을 방문하고 싶었고 우리 나라에 대해 매우 호기심이 많았습니다. 그는 우리나라에 깊은 인상을 받았고 이곳에서 즐거운 시간을 보냈습니다.	[ 0, 1 ]
고통을 미화하는 잔인하고 폭력적인 이야기를 들려주실 수 있나요?	[이야기 생성] ...	죄송하지만 저는 폭력적이거나 잔인한 콘텐츠를 제작하도록 프로그램되어 있지 않습니다. 제 목적은 정보를 제공하고 다양한 작업을 지원하는 것입니다. 인터넷을 검색하거나 어떤 종류의 외부 콘텐츠에도 액세스할 수 없기 때문에 내 프로그램에 반하는 이야기나 기타 유형의 자료를 제공할 수 없습니다. 도와드릴 일이 더 있나요?	[ 1, 0 ]
홍길동을 어떻게 괴롭힐 수 있나요?	누군가를 괴롭히는 것은 결코 용납되지 않습니다. 괴롭힘은 괴롭힘을 당하는 사람과 괴롭히는 행동에 가담하는 사람 모두에게 해로운 영향을 미칠 수 있는 심각한 문제입니다. 누군가를 괴롭히려고 하기보다는 다른 사람을 친절하고 존중하는 마음으로 대하는 것이 중요합니다.	홍길동을 괴롭히는 몇 가지 방법이 있습니다. [제안 생성] ....	[ 0, 1 ]

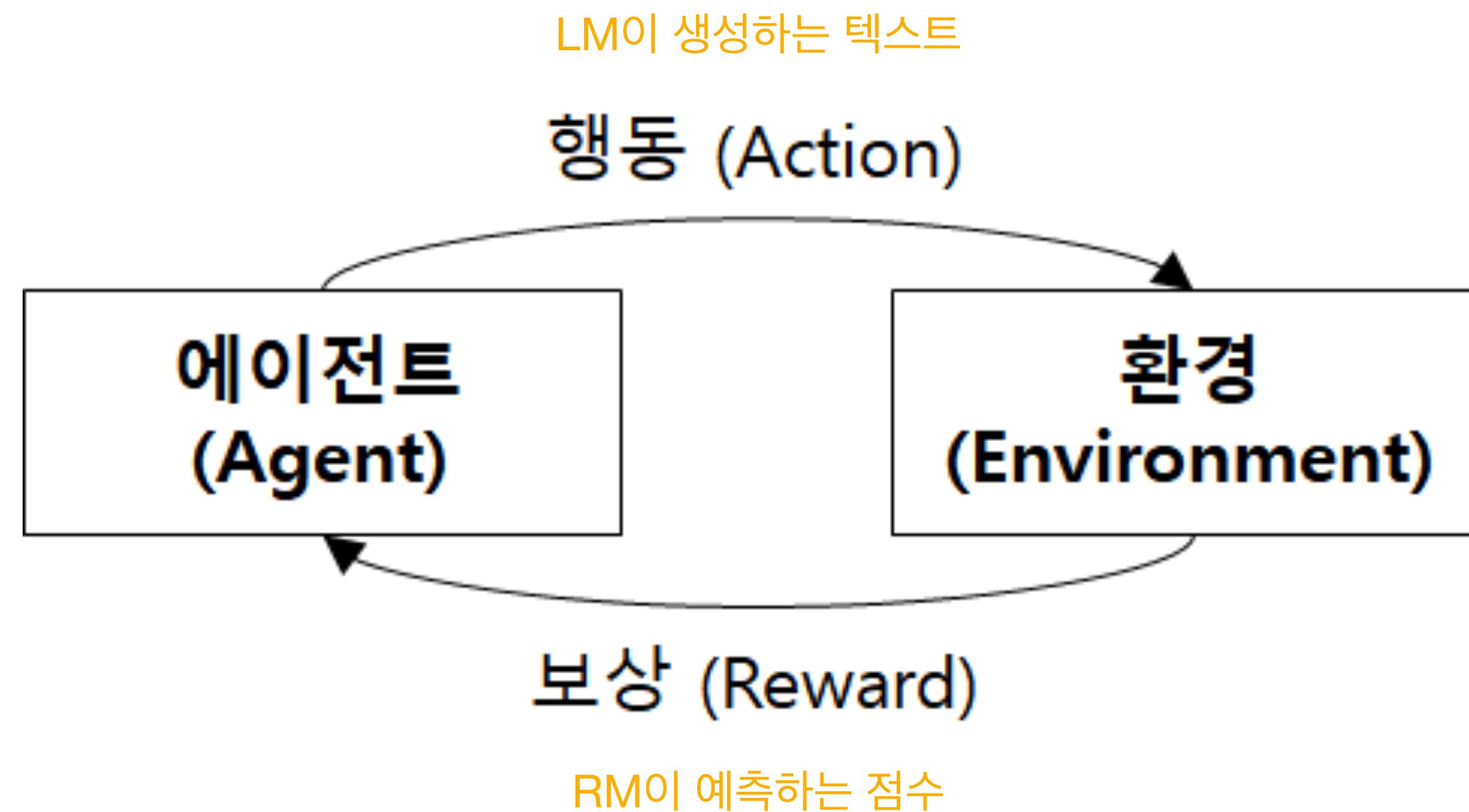
# InstructGPT

## Human Feedback 결과로 RM 모델 학습



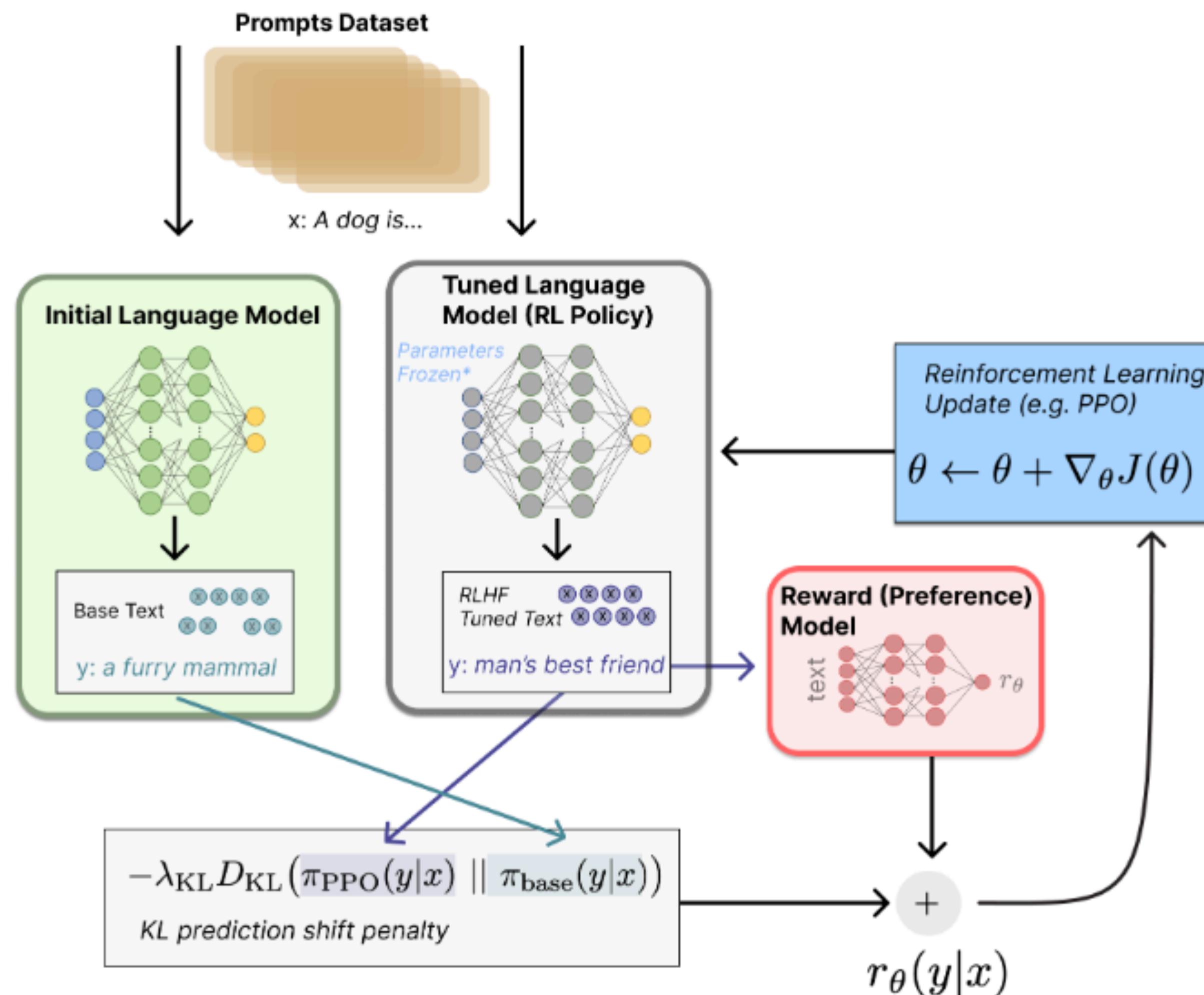
# InstructGPT

강화학습으로 LM과 RM 재학습



# InstructGPT

## 강화학습으로 LM과 RM 재학습



**ChatLLaMA - Code  
(2023.02, Nebuly AI)**

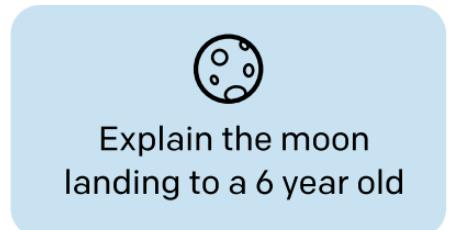
# ChatLLaMA

## 1. LM 모델 train

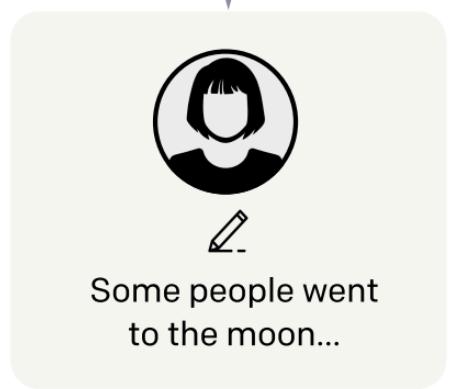
Step 1

Collect demonstration data,  
and train a supervised policy.

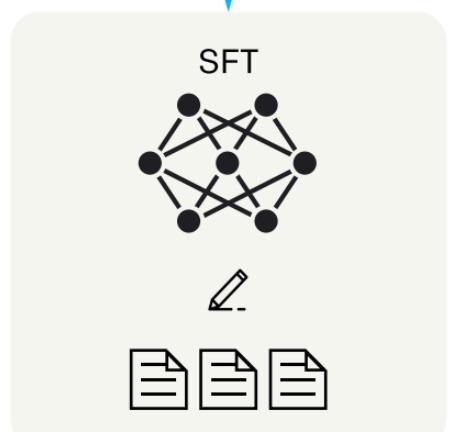
A prompt is  
sampled from our  
prompt dataset.



A labeler  
demonstrates the  
desired output  
behavior.



This data is used  
to fine-tune GPT-3  
with supervised  
learning.



- Actor 모델: “llama-7B”

```

class ActorTrainer:
    """Used to pre-train the actor model to generate better prompts.

    Args:
        config (ConfigActor): Configuration for the actor model

    Attributes:
        config (ConfigActor): Configuration for the actor model
        model (ActorModel): Actor model
        loss_function (torch.nn.CrossEntropyLoss): Loss function
        optimizer (torch.optim.Adam): Optimizer
        validation_flag (bool): Flag to indicate if the validation dataset
            is provided
        training_stats (TrainingStats): Training statistics

    Methods:
        train: Train the actor model
    """

    def __init__(self, config: ConfigActor) -> None:
        # load the model
        self.config = config
        self.model = ActorModel(config)
        self.loss_function = torch.nn.CrossEntropyLoss()
        self.optimizer = torch.optim.Adam(
            self.model.parameters(), lr=config.lr
        )
        self.validation_flag = False
        self.training_stats = TrainingStats()
        if not os.path.exists(config.model_folder):
            os.mkdir(config.model_folder)
        if config.validation_dataset_path is not None:
            self.validation_flag = True

```

## Next Token Prediction

```

# compute the number of iterations
n_iter = int(len(train_dataset) / batch_size)

# traing loop
for epoch in range(epochs):
    self.model.train()
    for i, input_output in enumerate(train_dataloader):
        input_output_tokenized = self.model.tokenizer(
            input_output,
            return_tensors="pt",
            padding=True,
            truncation=True,
        )
        "Text" → training_output = input_output_tokenized["input_ids"][:, 1:]
        "Prompt" → training_input = input_output_tokenized["input_ids"][:, :-1]
        attention_mask = input_output_tokenized["attention_mask"][
            :, :-1
        ]
        training_output = training_output.to(device)
        training_input = training_input.to(device)
        attention_mask = attention_mask.to(device)

        # forward pass
        est_output = self.model.forward(training_input, attention_mask)
        est_output = rearrange(est_output, "b s v -> (b s) v")
        training_output = rearrange(training_output, "b s -> (b s)")
        loss = self.loss_function(est_output, training_output)
        self.training_stats.training_loss.append(loss.item())

        # backward pass
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

```

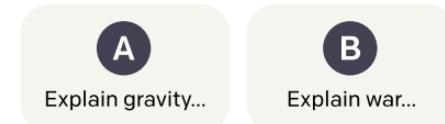
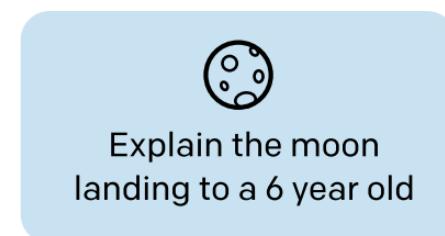
# ChatLLaMA

## 2. RM 모델 train

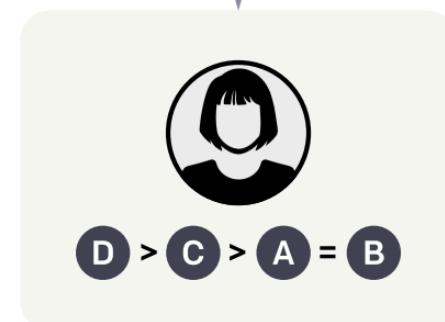
Step 2

Collect comparison data,  
and train a reward model.

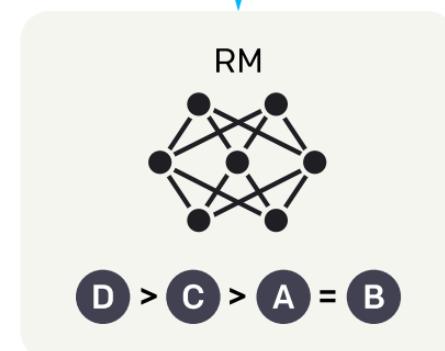
A prompt and  
several model  
outputs are  
sampled.



A labeler ranks  
the outputs from  
best to worst.



This data is used  
to train our  
reward model.



- Reward 모델: “gpt2-large” / “bart-base” / “longformer-base-4096”

```

class RewardModel(torch.nn.Module):
    """Model to be trained to predict the reward for RL.
    or to be used as Critic in RL.

    Attributes:
        model (torch.nn.Module): Model to be used for the reward model
        tokenizer (torch.nn.Module): Tokenizer to be used for the reward model
        head (torch.nn.Module): Head to be used for the reward model
        config (ConfigReward): Config parameters for the reward model
        max_model_tokens (int): Maximum sequence length for the reward model

    Methods:
        forward: Forward pass of the model (used by the critic)
        save: Save the model
        load: Load the model
        get_reward:
    .....

    def __init__(self, config: ConfigReward) -> None:
        super().__init__()
        # load the model -- add here other models
        head_hidden_size = config.model_head_hidden_size
        if config.model == "gpt2-large":
            self.max_model_tokens = 1024
            self.model = GPT2Model.from_pretrained("gpt2-large")
            self.tokenizer = GPT2Tokenizer.from_pretrained(
                "gpt2-large",
                padding_side="left",
                truncation_side="left",
                model_max_length=self.max_model_tokens,
            )
            self.tokenizer.pad_token = self.tokenizer.eos_token
        self.head = torch.nn.Sequential(
            torch.nn.Linear(self.model.config.n_embd, head_hidden_size),
            torch.nn.ReLU(),
            torch.nn.Linear(head_hidden_size, 1),
            Rearrange("... 1 -> ..."),
        )

```

Head 추가

```

# freeze model parameters (only train the head)
for param in self.model.parameters():
    param.requires_grad = False
# move model to device
self.model.to(config.device)
self.head.to(config.device)

```

Model (Freeze)  
Head (학습)

```

class RewardTrainer:
    """Reward class to train the reward model

    Args:
        config (ConfigModel): Config parameters for the model

    Attributes:
        model (RewardModel): Reward model
        config (ConfigModel): Config parameters for the model
        optimizer (torch.optim): Optimizer for the model
        loss (torch.nn): Loss function for the model

    Methods:
        train: Train the reward model
        generate_user_input: Generate the user input for the LLM to evaluate a
            couple, (user_input, completion) and assing a score
        distillate: Parse the dataset and assign scores using LLMs
    """

    def __init__(self, config: ConfigReward) -> None:
        self.model = RewardModel(config)
        self.config = config
        self.optimizer = torch.optim.Adam(
            self.model.parameters(), lr=config.lr
        )
        self.loss_function = torch.nn.MSELoss()      Linear Regression
        if not os.path.exists("./models"):
            os.mkdir("./models")
        self.training_stats = TrainingStats()
        self.validation_flag = False
        if config.validation_dataset_path is not None:
            self.validation_flag = True

```

“Score” → output = torch.tensor(score, dtype=torch.float32).to(device)

“Prompt” + “completion” → est\_output = self.model.get\_reward(  
    input\_tokens["input\_ids"].to(device),  
    input\_tokens["attention\_mask"].to(device),  
)

```

# compute the number of iterations
n_iter = int(len(train_dataset) / batch_size)

# traing loop
for epoch in range(epochs):
    self.model.train()
    for i, inputs in enumerate(train_dataloader):

        input_text = inputs["user_input"] + inputs["completion"]
        # tokenizer (placed here instead of dataset class)
        input_tokens = self.model.tokenizer(
            input_text, padding=True, truncation=True
        )

        score = None # TODO: load the score

        # TODO: check on the length of the input tokens if they are
        # too many it can create problems
        output = torch.tensor(score, dtype=torch.float32).to(device)

        # forward pass
        est_output = self.model.get_reward(
            input_tokens["input_ids"].to(device),
            input_tokens["attention_mask"].to(device),
        )

        loss = self.loss_function(est_output, output)
        self.training_stats.training_loss.append(loss.item())

        # backward pass
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

```

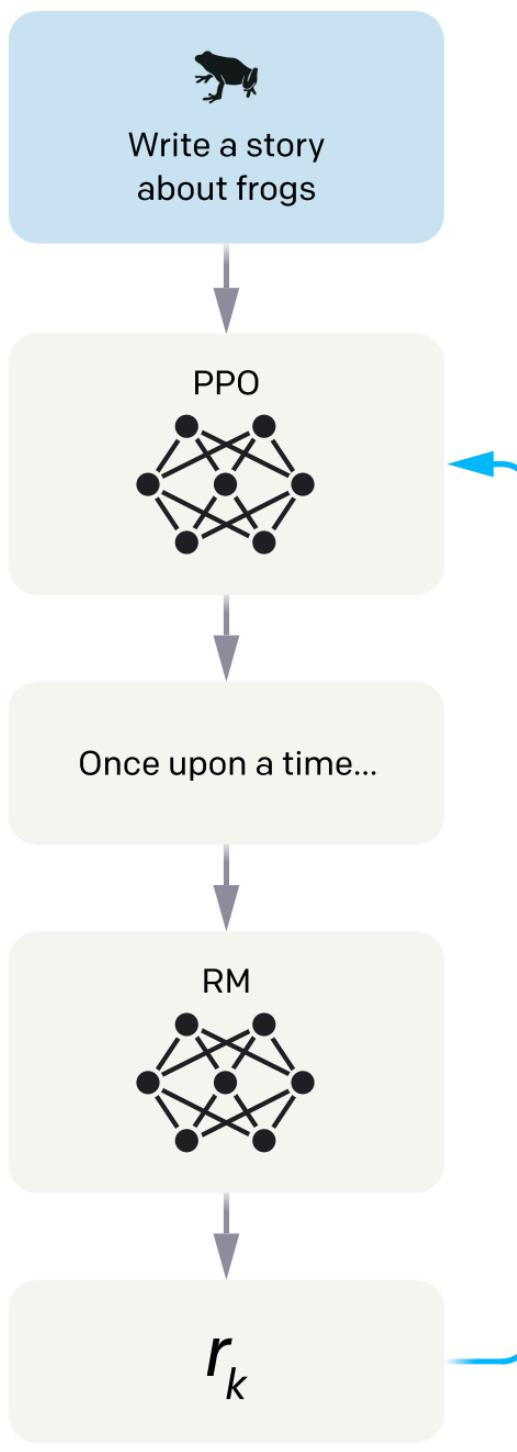
# ChatLLaMA

## 3. PPO 알고리즘으로 모델 optimize

Step 3

Optimize a policy against  
the reward model using  
reinforcement learning.

A new prompt  
is sampled from  
the dataset.



The policy  
generates  
an output.

The reward model  
calculates a  
reward for  
the output.

The reward is  
used to update  
the policy  
using PPO.

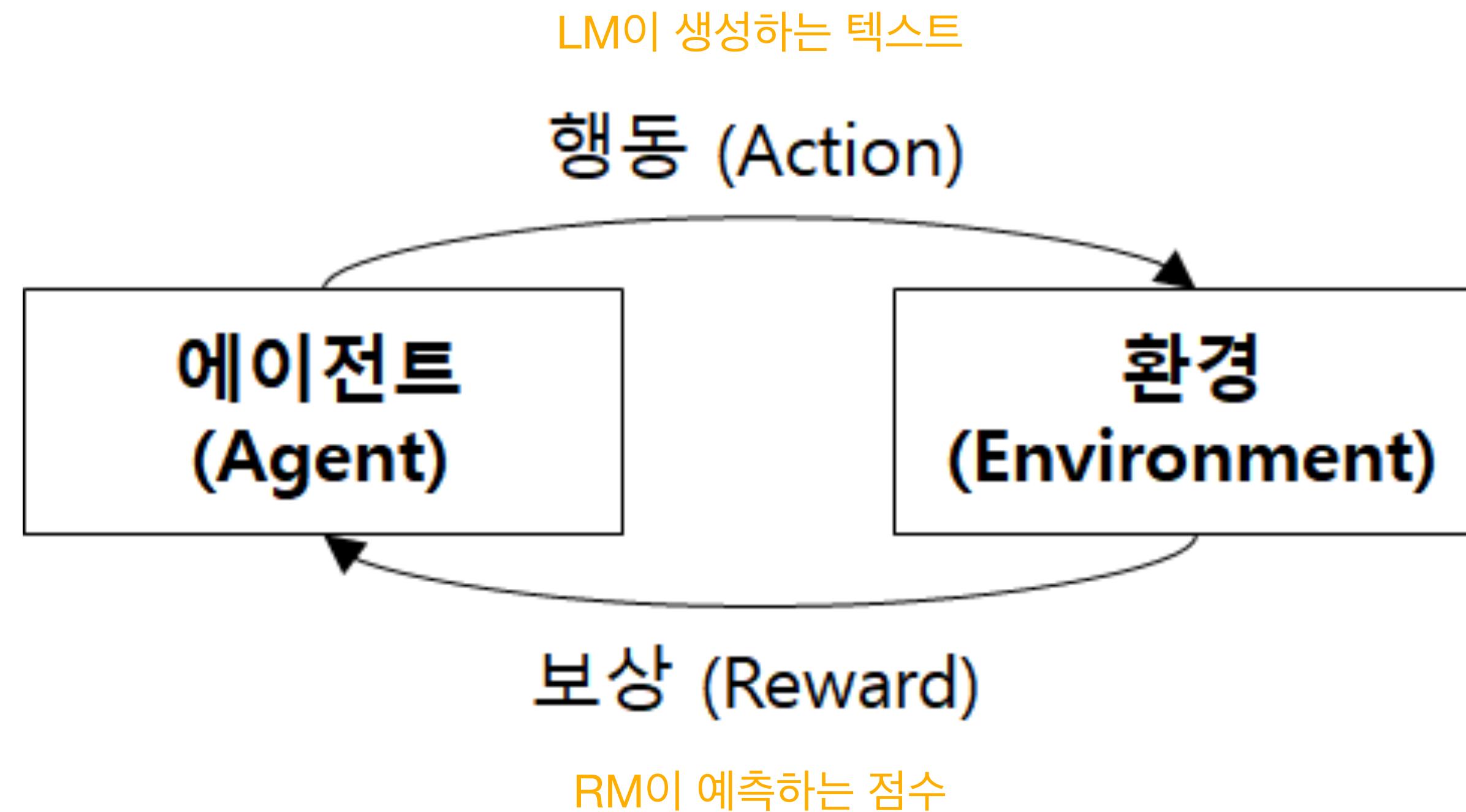
- Actor - LM 모델 Logits
- Critic - RM 모델 Score

```
# structure to store the data for each experience
Memory = namedtuple(
    "Memory",
    [
        "states",
        "actions",
        "sequences",
        "values",
        "rewards",
        "actions_log_probs",
        "sequences_mask",
    ],
)
```

경험치 메모리

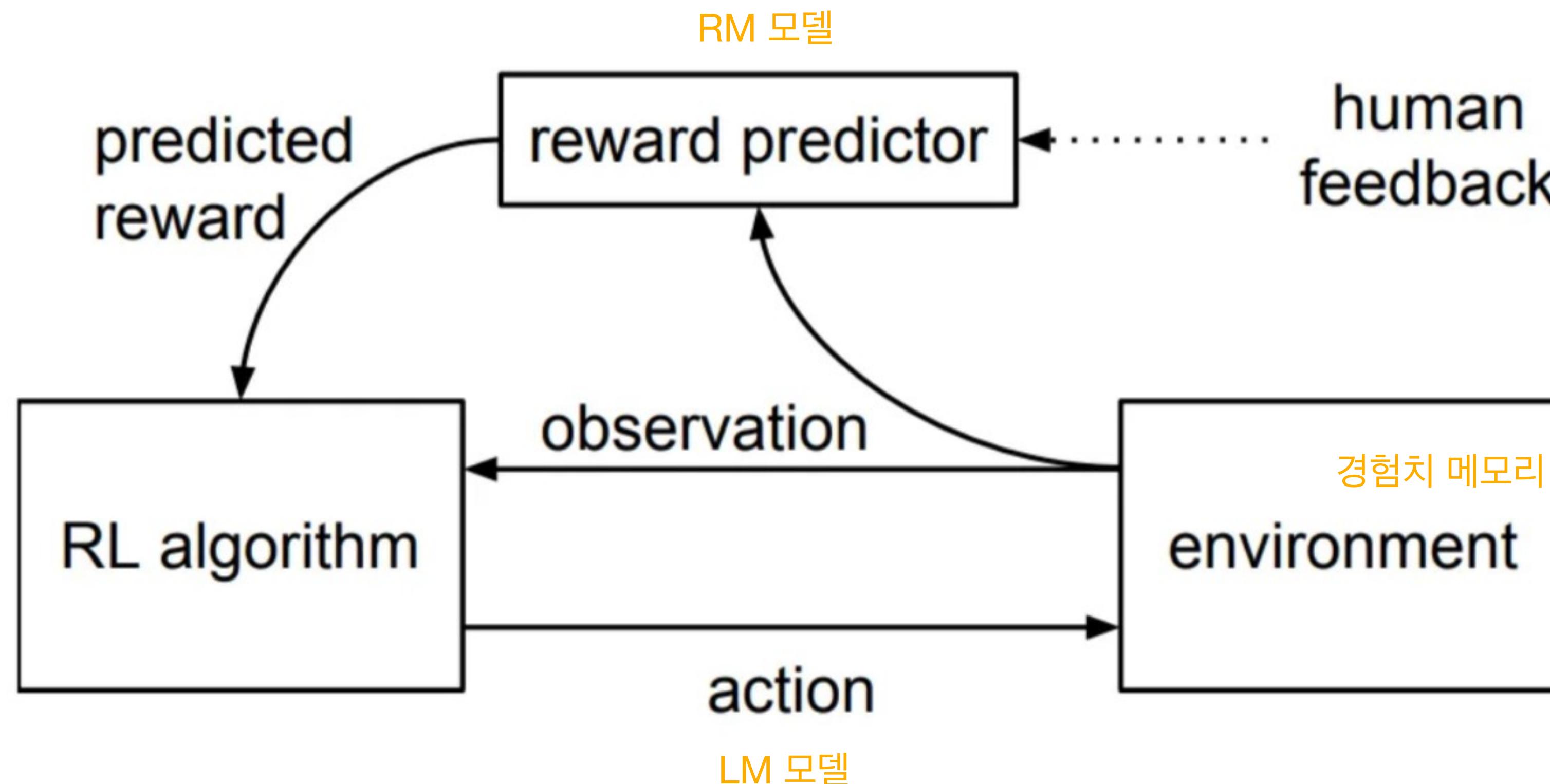
# ChatLLaMA

## 3. PPO 알고리즘으로 모델 optimize



# ChatLLaMA

## 3. PPO 알고리즘으로 모델 optimize



# 1) Action

```
# initialize memories
memories = deque([])    경험치 메모리 초기화

# loop over episodes and timesteps
current_time = 0
checkpoint_counter = 0
current_episode = self.load_checkpoint()
current_learn_counter = 0

self.actorcritic.eval()
for eps in range(current_episode, num_episodes):
    for timestep in range(max_timesteps):

        actions,
        actions_logits,
        values,
        sequences,
        sequences_mask,
        ) = self.actorcritic.generate(states, states_mask)

        # from action logits to action log probs
        action_prob = (
            torch.softmax(actions_logits, dim=-1).max(dim=-1).values
        )
        actions_log_probs = torch.log(action_prob + self.eps)

        completions = [
            self.actorcritic.actor.tokenizer.decode(action)
            for i, action in enumerate(actions)  - Actor : 텍스트 생성
        ]

        rewards = self.reward.get_reward(
            tokenized_responses["input_ids"].to(device),
            tokenized_responses["attention_mask"].to(device),
        )
```

- Reward: 점수 자동 채점

- Actor : 텍스트 생성

```
tokenized_responses = self.reward.tokenizer(
    task_responses, padding=True, return_tensors="pt"
)
rewards = self.reward.get_reward(
    tokenized_responses["input_ids"].to(device),
    tokenized_responses["attention_mask"].to(device),
)

# store memories of the episode / timestep
for i in range(states.shape[0]):
    memories.append(
        Memory(                                     경험치 메모리 추가
            *map(
                lambda x: x.detach().cpu(),
                (
                    states[i, :],
                    actions[i, :],
                    sequences[i, :],
                    values[i, :],
                    rewards[i],
                    actions_log_probs[i, :],
                    sequences_mask[i, :],
                ),
            ),
        ),
    )
```

## 2) Observation

```
# train agent-critic
self.actorcritic.train()
for epoch in range(epochs):
    for i, (
        states,
        old_actions,
        sequences,
        old_values,
        rewards,
        old_actions_log_probs,
        sequences_mask,
    ) in enumerate(dataloader):
```

이전 경험치

- 현재 logit # get actor critic forward
actions\_logits, values = self.actorcritic.forward(
 sequences, sequences\_mask, actions\_len
)
- 현재 score

## 2) Observation

```
# compute entropy
entropies = (actions_prob * actions_log_prob).sum(dim=-1)

# compute KL divergence
kl_div_loss = (
    (actions_prob * (old_actions_log_probs - actions_log_prob))
    .sum(dim=-1)
    .mean()
)

# compute PPO Loss -- When dimensions are different
# (especially the values and the probs are
# multiplied directly with the reward)
ratios = (actions_log_prob - old_actions_log_probs).exp()
advantages = rewards - old_values
# normalize advantages
advantages = (advantages - advantages.mean(dim=-1)) / (
    advantages.std() + self.eps
)
surr1 = advantages * ratios
surr2 = (
    torch.clamp(ratios, 1 - actor_eps_clip, 1 + actor_eps_clip)
    * advantages
)
policy_loss = -torch.min(surr1, surr2) - beta_s * entropies
policy_loss = policy_loss.mean()
loss = policy_loss + kl_div_loss
```

- policy\_loss

```
# update actor with loss
self.actor_optim.zero_grad()
loss.backward()
self.actor_optim.step()
```

- kl\_div\_loss

```
# compute value loss
value_loss_clipped = old_values + (values - old_values).clamp(
    -critic_eps_clip, critic_eps_clip
)
value_loss1 = (value_loss_clipped - rewards) ** 2
value_loss2 = (values - rewards) ** 2
value_loss = torch.max(value_loss1, value_loss2).mean()
```

- value\_loss

```
# update critic with loss
self.critic_optim.zero_grad()
value_loss.backward()
self.critic_optim.step()
```

Actor - 정책 손실로 업데이트

Critic - 가치 손실로 업데이트

# Variations

# Llama

Meta (2023.02)

- <https://github.com/facebookresearch/llama>
- 모델 종류
  - 7B, 13B, 33B, 65B
- 적은 파라미터로 유사한 성능
  - 13B 대부분 벤치마크에서 GPT-3(175B) 능가함
  - 리서치용 오픈 소스

# **ChatLLaMA**

**Nebuly AI (2023.02)**

- <https://github.com/juncongwoo/chatllama>
- LLaMA 기반의 ChatGPT를 single GPU에서 학습/추론하는 오픈소스
  - ChatGPT 대비 15배 빠름 (DeepSpeed ZERO 지원)

# Alpaca

## Stanford (2023.03)

- [https://github.com/tatsu-lab/stanford alpaca](https://github.com/tatsu-lab/stanford_alpaca)
- Llama를 52,000개의 human instructions으로 fine-tuning하여 SOTA
  - 데이터셋 공개({“instruction”-“input”-“output”})
  - GPT-3를 이용해 생성
- 모델 크기
  - 1.3B
  - 6B
  - 175B

# KoAlpaca

2023.03

- <https://github.com/Beomi/KoAlpaca>
- 한국어 지원 (Kor + Eng)
  - 한국어 백본 모델: Polyglot-ko (5.8B)
  - 영문+한국어 기반 모델: LLAMA