

Large Language Model

OpenAI API, LangChain and
Finetuning LLM

조준우

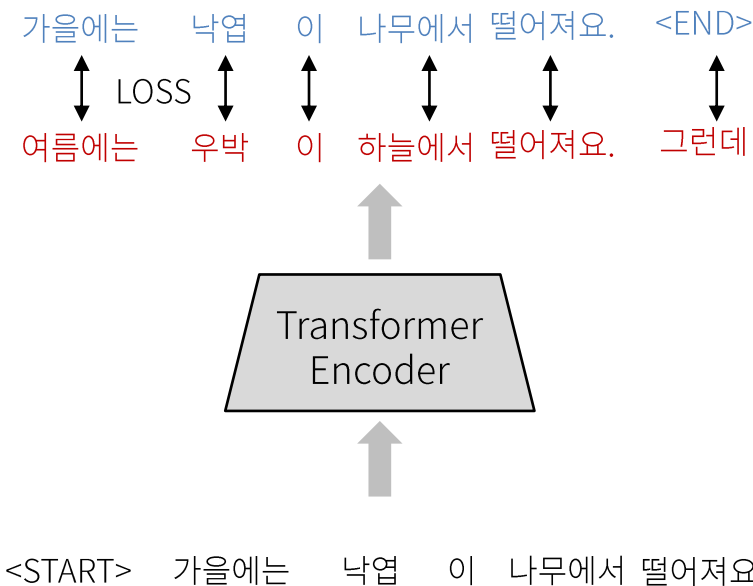
metamath@gmail.com

A diagram showing a hexagonal lattice. A central hexagon is shaded light gray and contains the symbol $\partial \mathbf{x}$. To its left, another hexagon is shaded light gray and contains the symbol \log . Above the \log hexagon is a white hexagon containing the symbol Σ . Below the \log hexagon is a white hexagon containing a black dot and a curved line. A yellow horizontal bar is located below the bottom row of hexagons.

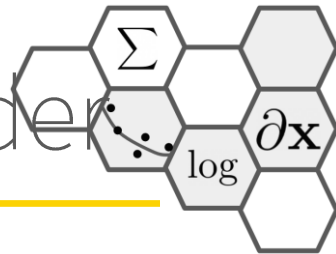
- 텍스트

- <START>가을에는 낙엽이 나무에서 떨어져요.<END>
- 입력: <START>가을에는 낙엽이 나무에서 떨어져요.
- 타겟: 가을에는 낙엽이 나무에서 떨어져요.<END>

- 이전 단어(토큰)를 보고 다음 토큰을 예측하는 능력을 배울 수 있음

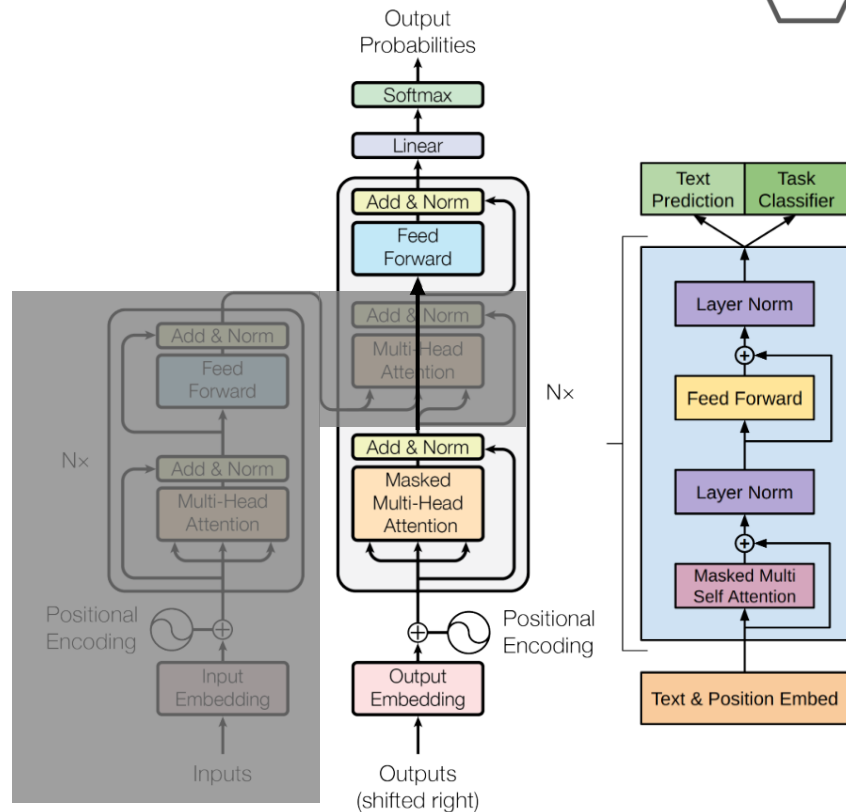


Model: GPT; Transformer Decoder

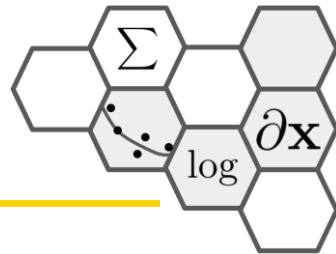


- Generative Pre-Training
 - Transformer에서 Decoder 부분만 사용한 언어 모델
 - Cross Attention 제외
 - Auto-Regressive
- 언어 모델 Language Model
 - 앞 단어를 보고 뒤 단어를 예측

Translate English to Korean: Soccer →

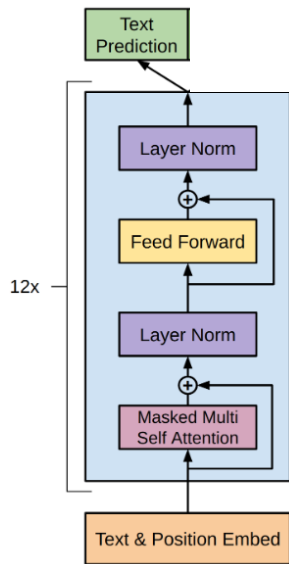


Model: GPT

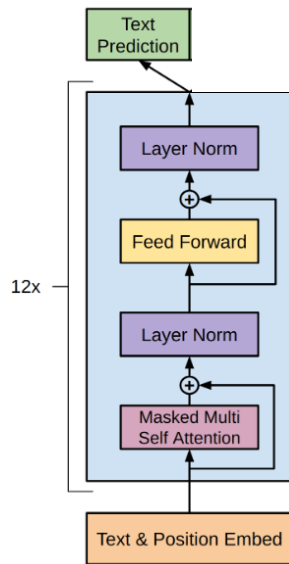


- Auto Regressive

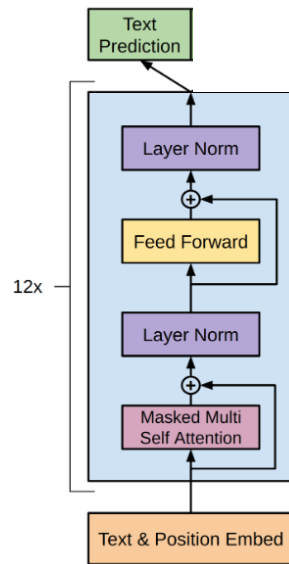
I thought this movie **was**



I thought this movie was **great**



I thought this movie was great **and**

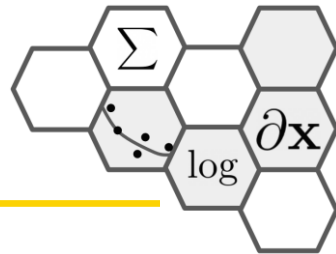


<START>I thought this movie

<START>I thought this movie **was**

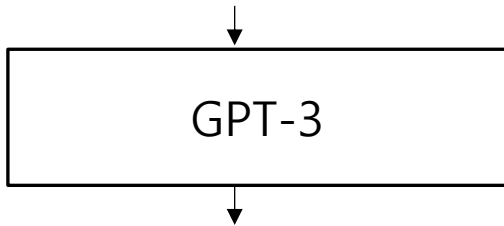
<START>I thought this movie was **great**

Model: GPT-3



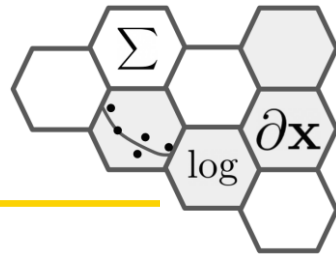
- In Context Learning
 - 32 sequences

Translate English to Korean : \n I love soccer [tokorean] 나는 축구를 좋아해 \n
I have to go to school [tokorean] 나는 학교에 가야만 해 \n [tokorean] I live in
Chicago [tokorean]



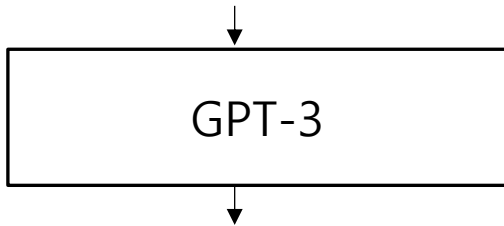
Output1 Output2 ... Output31 나는

Model: GPT-3



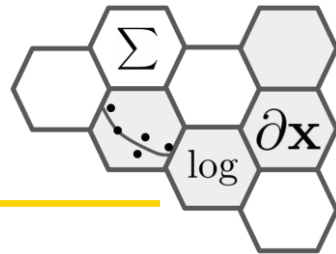
- In Context Learning
 - 32 sequences

Translate English to Korean : \n I love soccer [tokorean] 나는 축구를 좋아해 \n
I have to go to school [tokorean] 나는 학교에 가야만 해 \n [tokorean] I live in
Chicago [tokorean] 나는



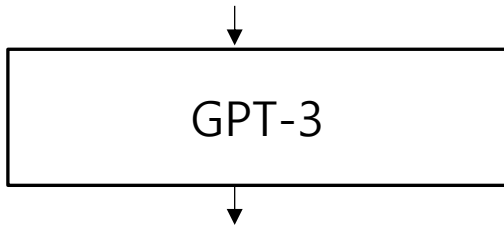
Output1 Output2 ... Output31 Output32 시카고에

Model: GPT-3



- In Context Learning
 - 32 sequences

Translate English to Korean : \n I love soccer [tokorean] 나는 축구를 좋아해 \n
I have to go to school [tokorean] 나는 학교에 가야만 해 \n [tokorean] I live in
Chicago [tokorean] 나는 시카고에



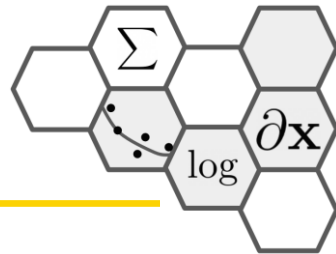
Output1 Output2 ... Output31 Output32 Output33 산다

A diagram showing a hexagonal lattice structure. A central hexagon is shaded light gray and contains the symbol $\partial \mathbf{x}$. To its left, another hexagon is shaded light gray and contains the symbol \log . Above the \log hexagon is a white hexagon containing the symbol Σ . Below the \log hexagon is a white hexagon containing a curved line with three dots. A yellow horizontal bar is positioned below the bottom row of hexagons.

- Q: 프랑스의 수도는?

GPT-3 175B Parameters

Instruction Fine-Tuning



- Limitation of In-Context Learning

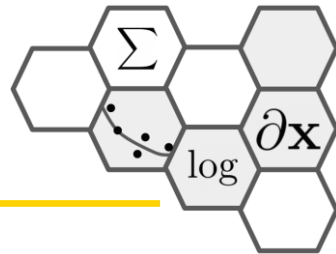


Even with
multiple
examples

- In-context learning may not work for smaller models **LLM**
- Examples take up space in the context window

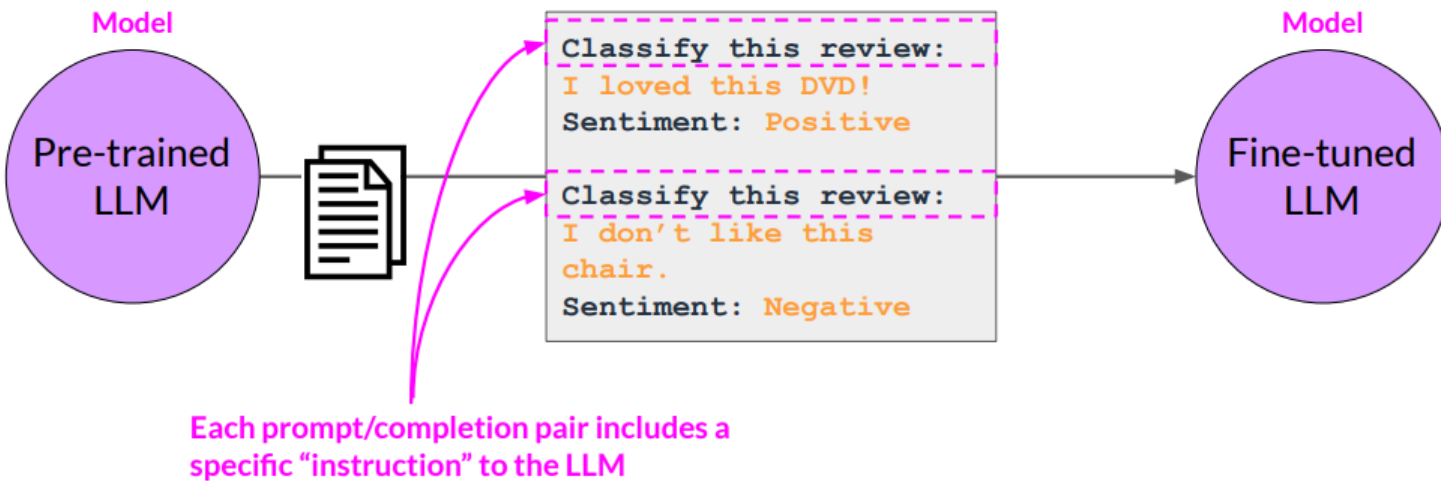
Instead, try **fine-tuning** the model

Instruction Fine-Tuning

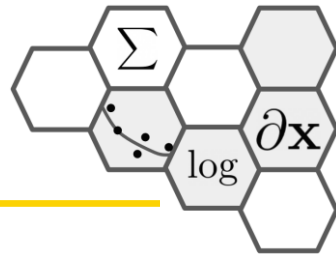


- Using prompts to fine-tune LLMs with instruction

LLM fine-tuning

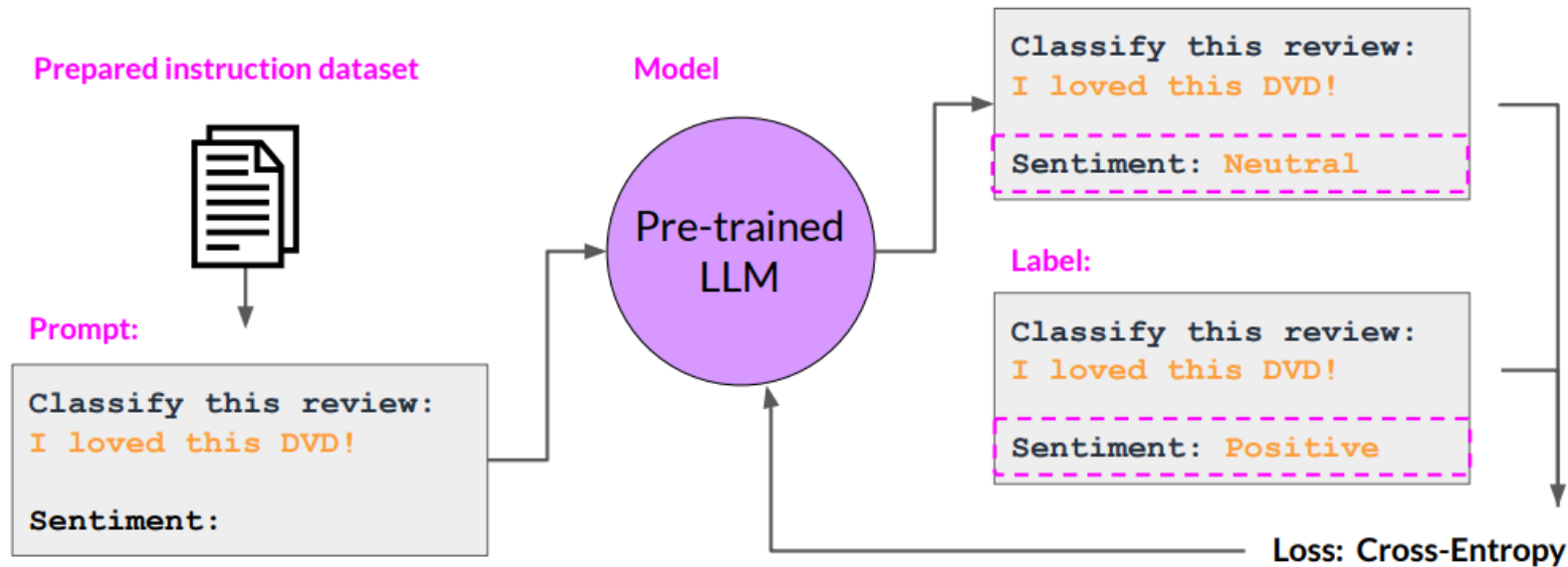


Instruction Fine-Tuning

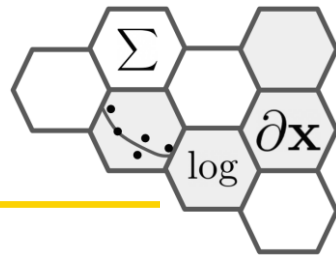


- Fine-tuning process

LLM fine-tuning



Model: InstructGPT



- RLHF: Reinforcement Learning from Human Feedback

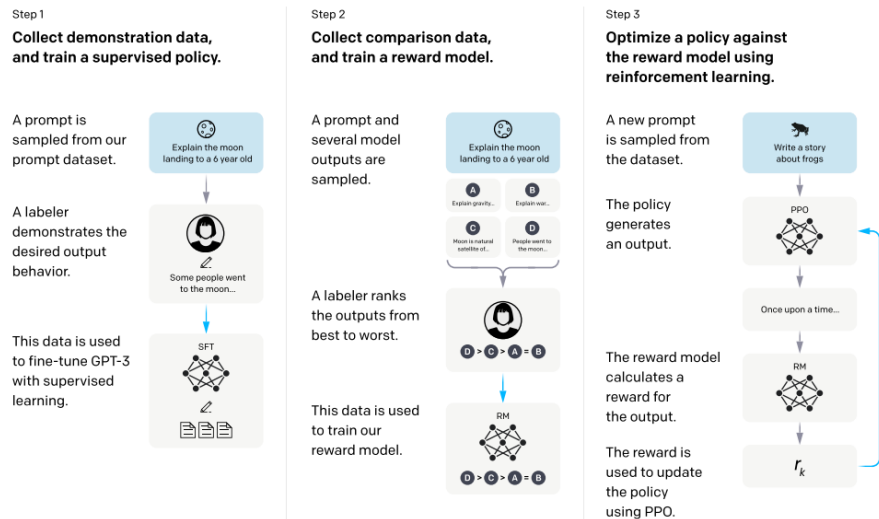
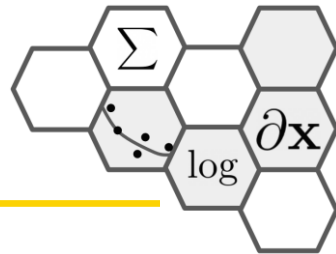
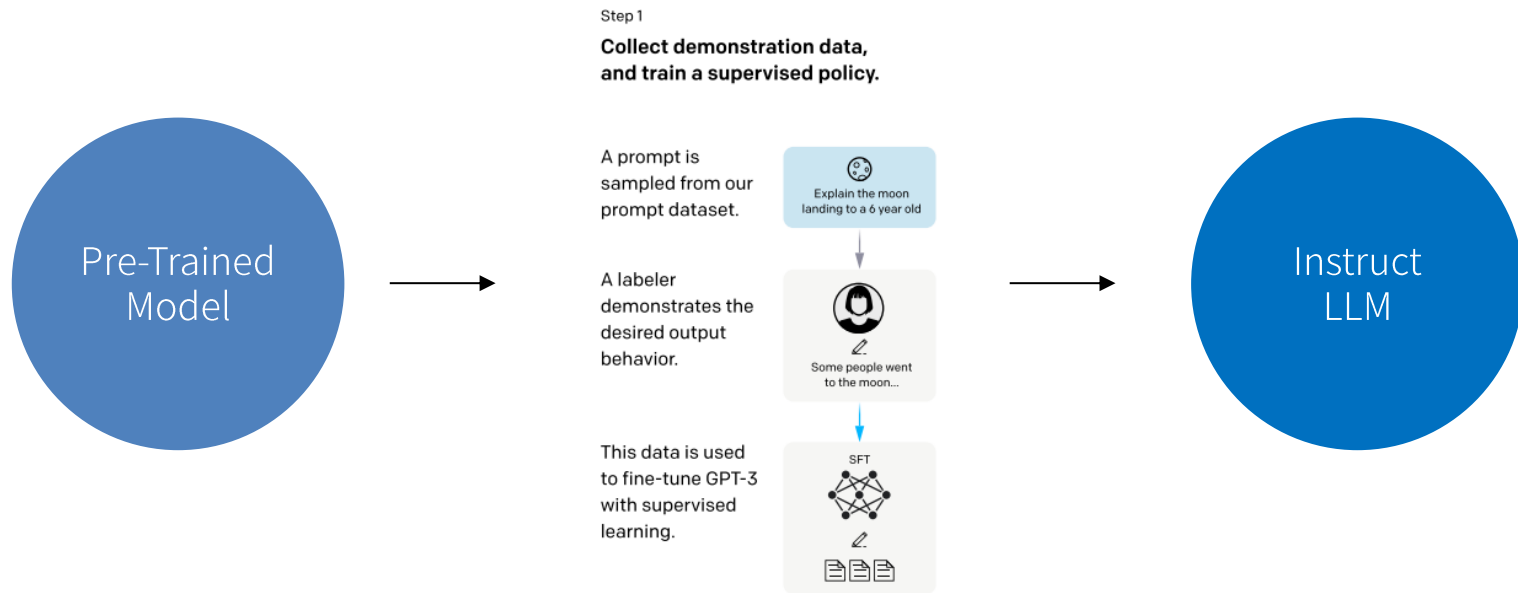


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

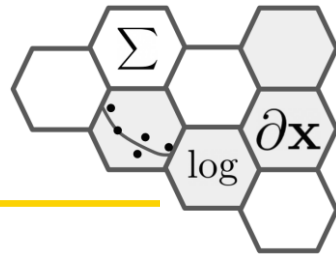
Model: InstructGPT



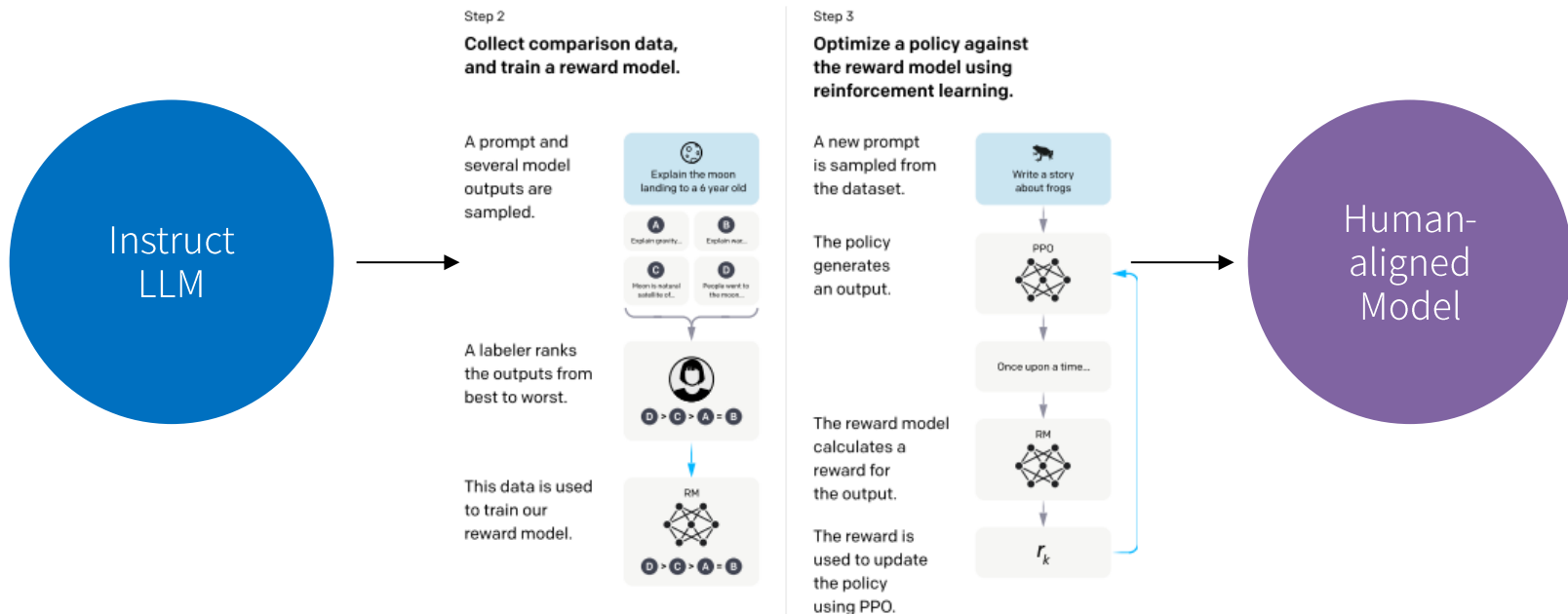
- SFT: Supervised Fine Tuning



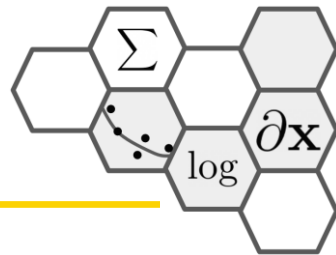
Model: InstructGPT



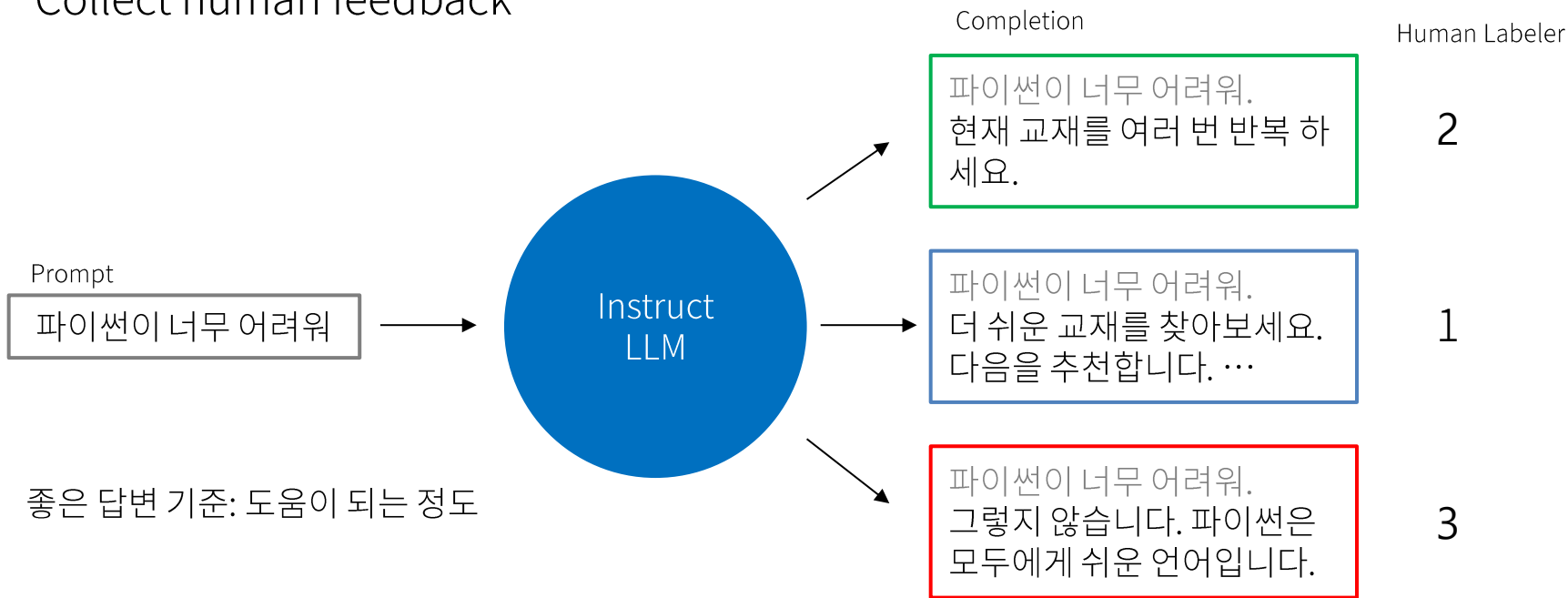
- Training Reward Model and Reinforcement Learning



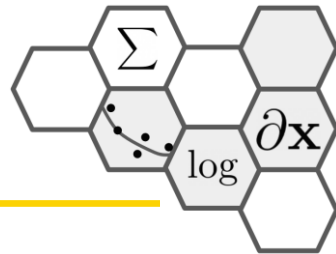
Reward Model



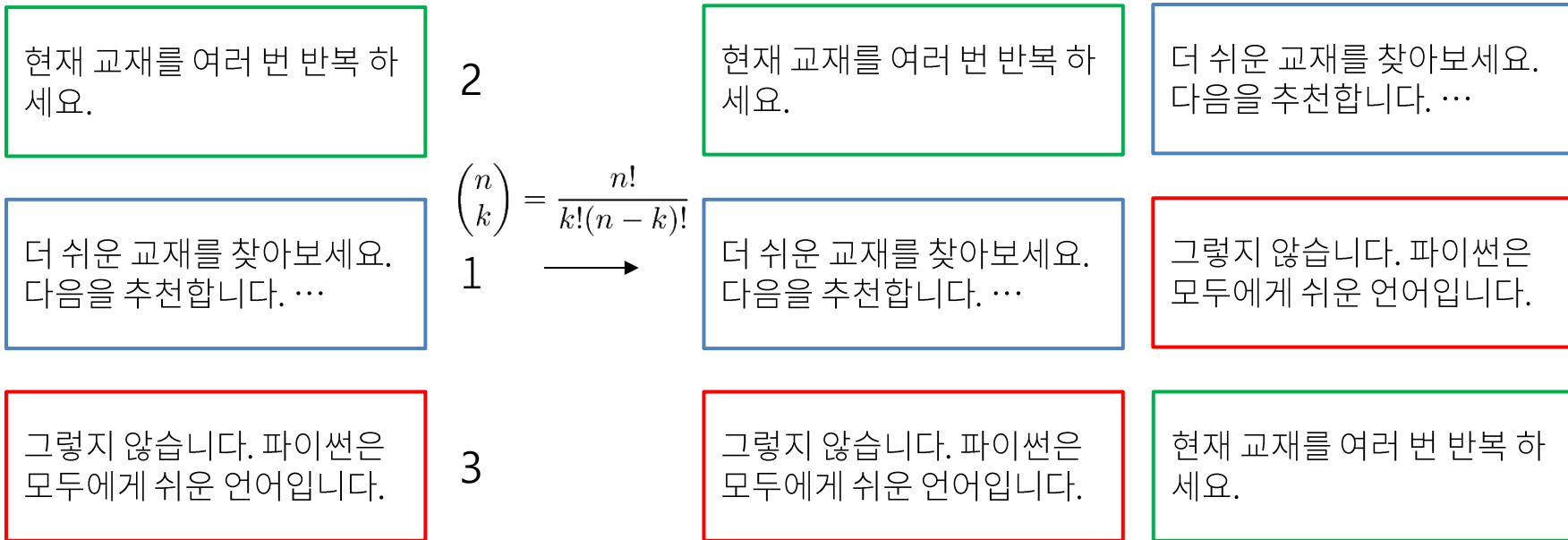
- Collect human feedback



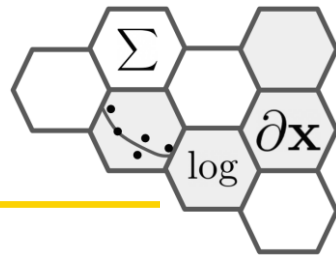
Reward Model



- Prepare labeled data for training



Reward Model



- Train reward model

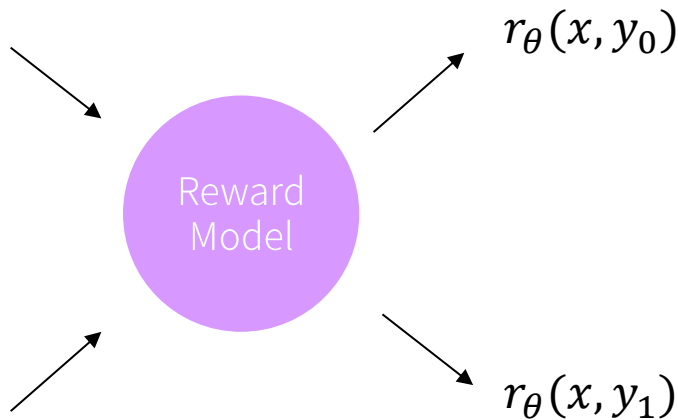
$$\text{loss}(r_\theta) = -E_{(x, y_0, y_1, i) \sim D} \left[\log \left(\sigma(r_\theta(x, y_i) - r_\theta(x, y_{1-i})) \right) \right]$$

x 파이썬이 너무 어려워

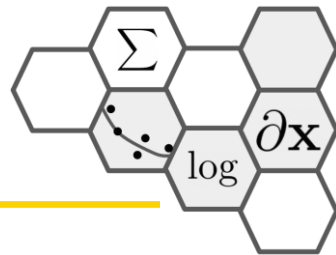
y_0 파이썬이 너무 어려워.
현재 교재를 여러 번 반복 하
세요.

x 파이썬이 너무 어려워

y_1 파이썬이 너무 어려워.
더 쉬운 교재를 찾아보세요.
다음을 추천합니다. ...



Reward Model



- Train reward model

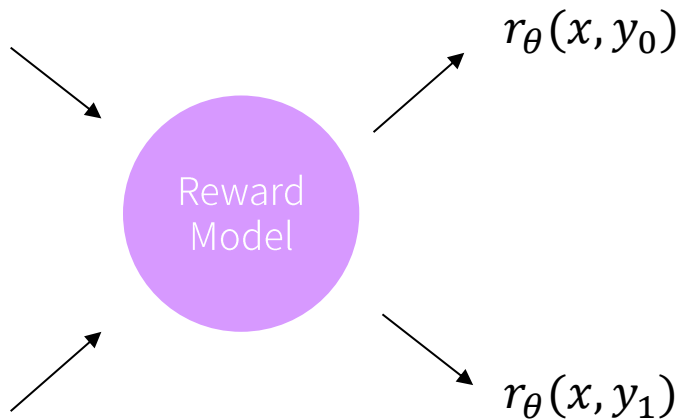
$$\text{loss}(r_\theta) = -E_{(x, y_0, y_1, i=1) \sim D} \left[\log \left(\sigma(r_\theta(x, y_1) - r_\theta(x, y_0)) \right) \right]$$

x 파이썬이 너무 어려워

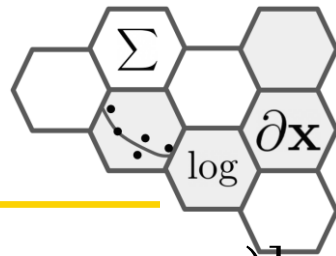
y_0 파이썬이 너무 어려워.
현재 교재를 여러 번 반복 하
세요.

x 파이썬이 너무 어려워

y_1 파이썬이 너무 어려워.
더 쉬운 교재를 찾아보세요.
다음을 추천합니다. ...

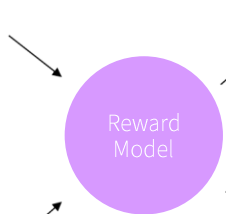


Reward Model



- Train reward model $loss(r_\theta) = -E_{(x, y_0, y_1, i=1) \sim D} [\log(\sigma(r_\theta(x, y_1) - r_\theta(x, y_0)))]$

x 파이썬이 너무 어려워
 y_0 파이썬이 너무 어려워.
 현재 교재를 여러 번 반복 하
 세요.



$r_\theta(x, y_0) = -1$

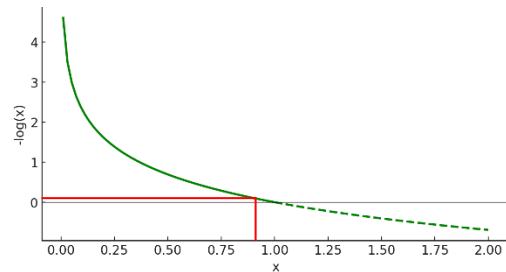
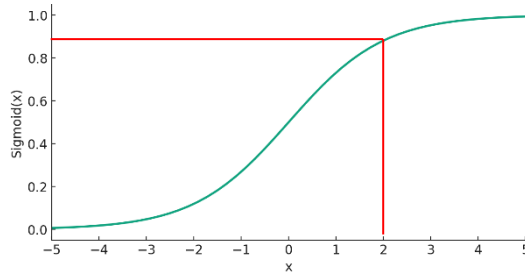
GOOD

$r_\theta(x, y_1)$ $r_\theta(x, y_0)$
 1 -1

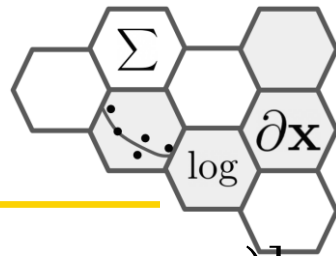
$r_\theta(x, y_1) - r_\theta(x, y_0)$
 2

x 파이썬이 너무 어려워
 y_1 파이썬이 너무 어려워.
 더 쉬운 교재를 찾아보세요.
 다음을 추천합니다. ...

$r_\theta(x, y_1) = 1$

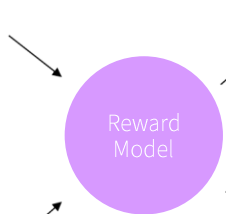


Reward Model



- Train reward model $loss(r_\theta) = -E_{(x, y_0, y_1, i=1) \sim D} [\log(\sigma(r_\theta(x, y_1) - r_\theta(x, y_0)))]$

x 파이썬이 너무 어려워
 y_0 파이썬이 너무 어려워.
 현재 교재를 여러 번 반복 하
 세요.



$r_\theta(x, y_0) = 1$

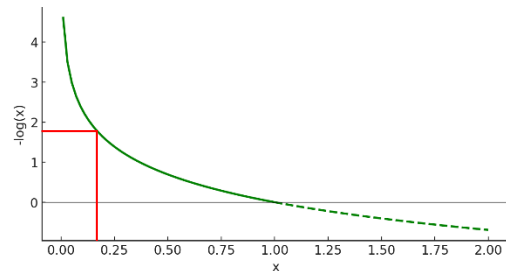
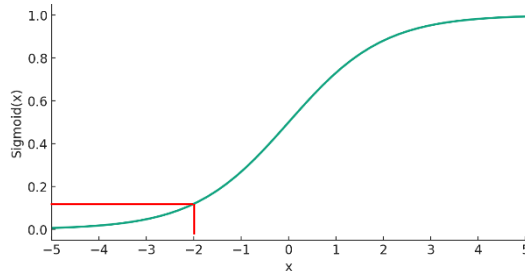
BAD

$r_\theta(x, y_1)$ $r_\theta(x, y_0)$
 -1 1

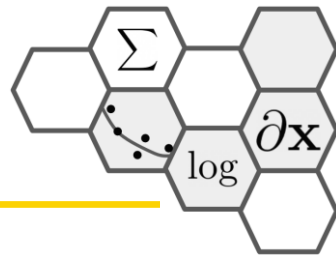
$r_\theta(x, y_1) - r_\theta(x, y_0)$
 -2

x 파이썬이 너무 어려워
 y_1 파이썬이 너무 어려워.
 더 쉬운 교재를 찾아보세요.
 다음을 추천합니다. ...

$r_\theta(x, y_1) = -1$

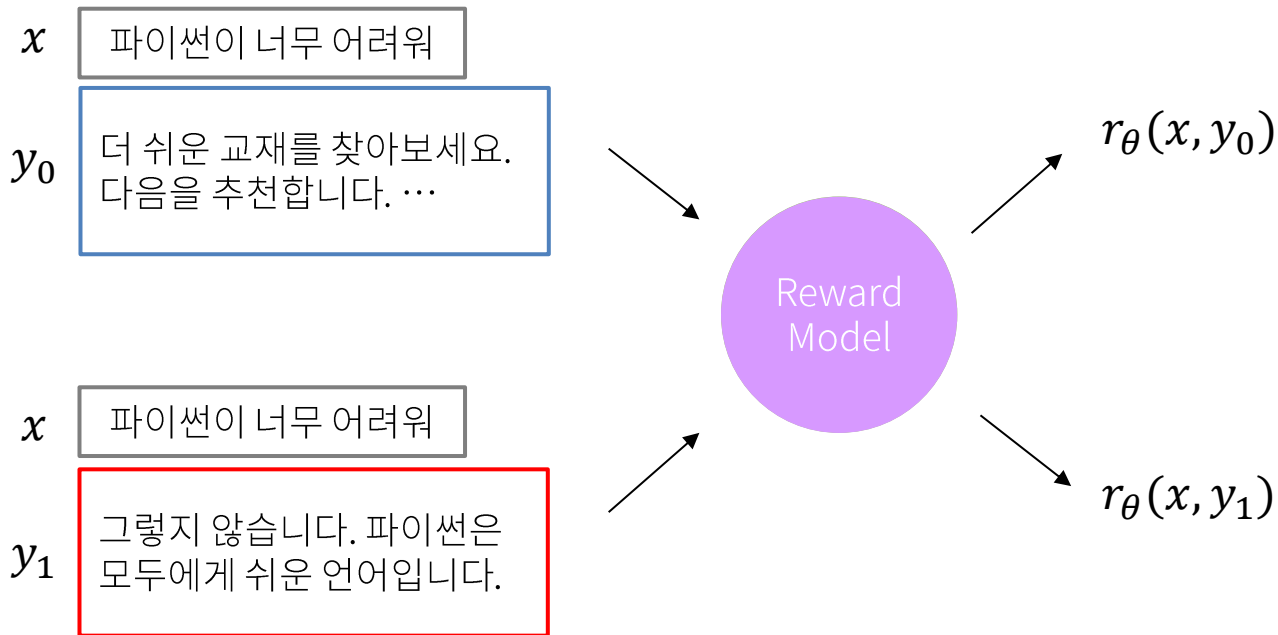


Reward Model

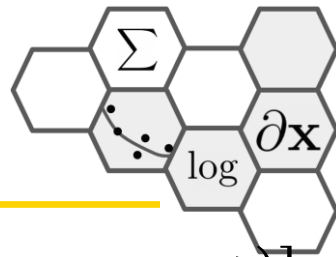


- Train reward model

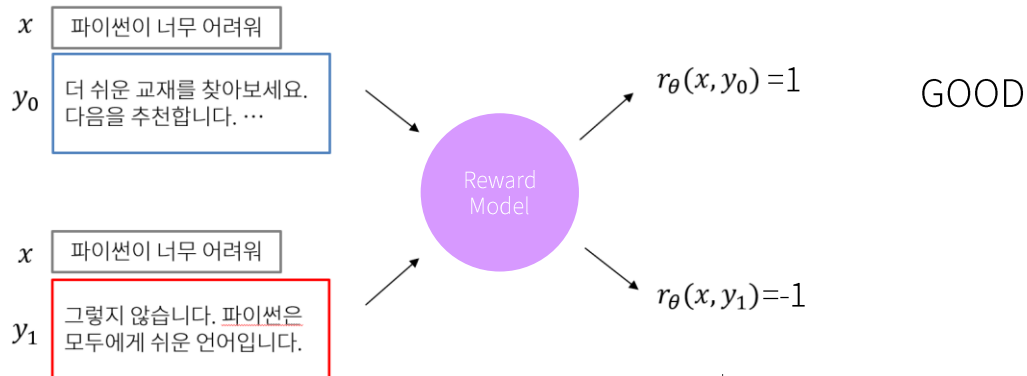
$$\text{loss}(r_\theta) = -E_{(x, y_0, y_1, i=0) \sim D} \left[\log \left(\sigma(r_\theta(x, y_0) - r_\theta(x, y_1)) \right) \right]$$



Reward Model

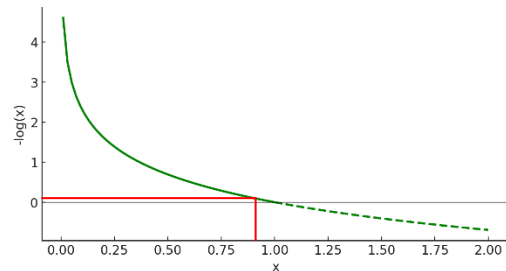
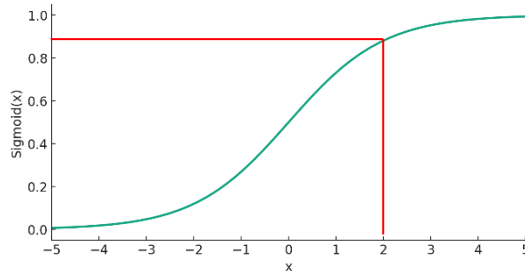


- Train reward model $loss(r_\theta) = -E_{(x, y_0, y_1, i=0) \sim D} [\log(\sigma(r_\theta(x, y_0) - r_\theta(x, y_1)))]$

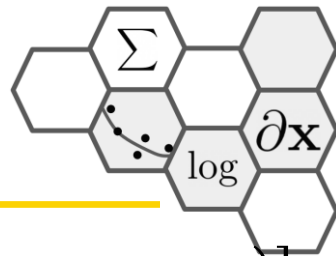


$$\begin{matrix} r_\theta(x, y_0) & r_\theta(x, y_1) \\ 1 & -1 \end{matrix}$$

$$\begin{matrix} r_\theta(x, y_0) - r_\theta(x, y_1) \\ 2 \end{matrix}$$

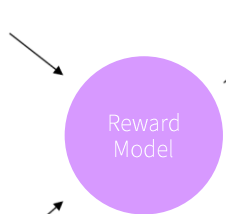


Reward Model



- Train reward model $loss(r_\theta) = -E_{(x, y_0, y_1, i=0) \sim D} [\log(\sigma(r_\theta(x, y_0) - r_\theta(x, y_1)))]$

x 파이썬이 너무 어려워
 y_0 더 쉬운 교재를 찾아보세요.
 다음을 추천합니다. ...



$r_\theta(x, y_0) = -1$

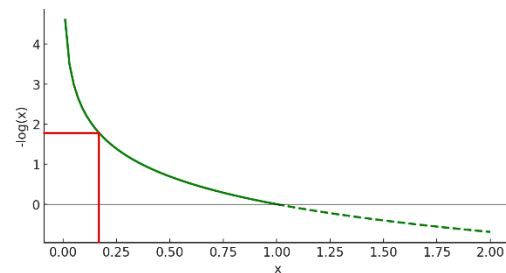
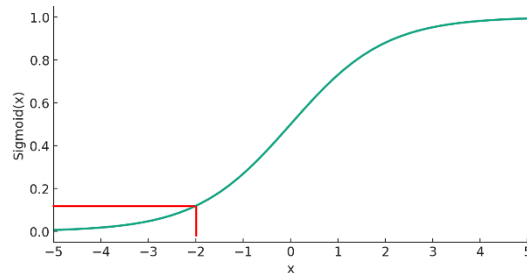
BAD

$r_\theta(x, y_0)$ $r_\theta(x, y_1)$
 -1 1

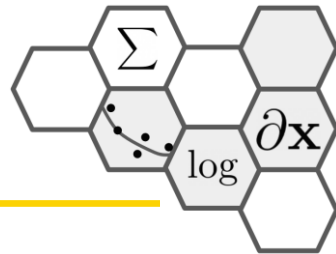
$r_\theta(x, y_0) - r_\theta(x, y_1)$
 -2

x 파이썬이 너무 어려워
 y_1 그렇지 않습니다. 파이썬은
 모두에게 쉬운 언어입니다.

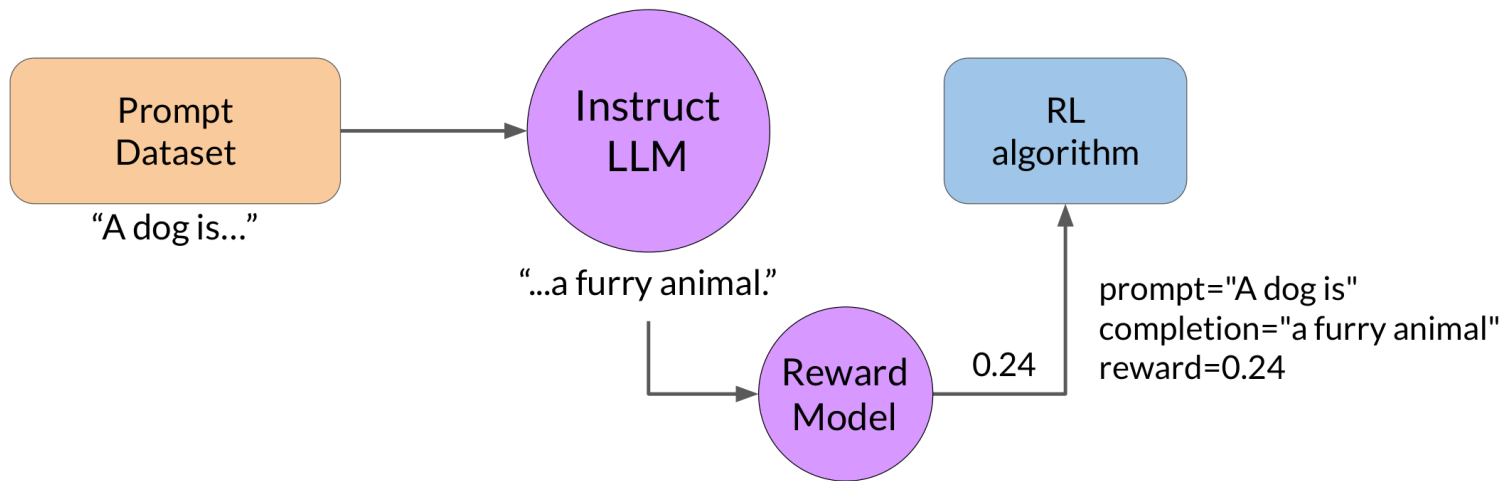
$r_\theta(x, y_1) = 1$



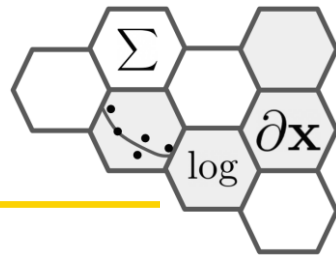
Fine-tuning with RLHF



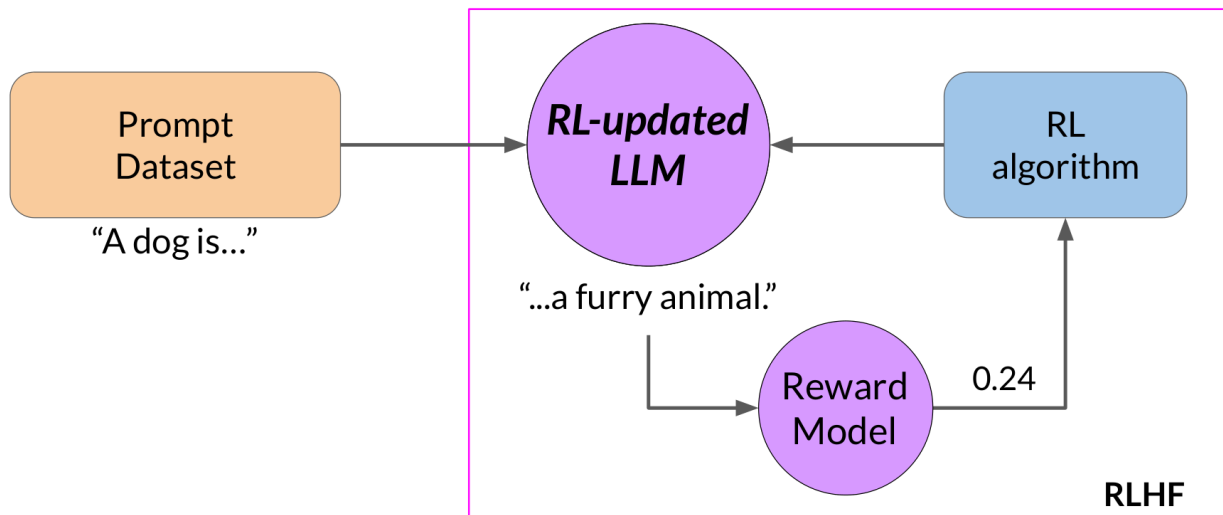
- Use the reward model to fine-tune LLM with RL



Fine-tuning with RLHF

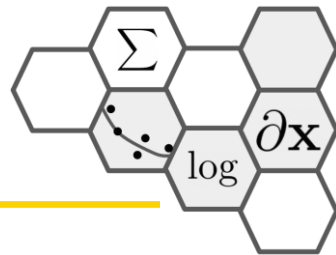


- Use the reward model to fine-tune LLM with RL

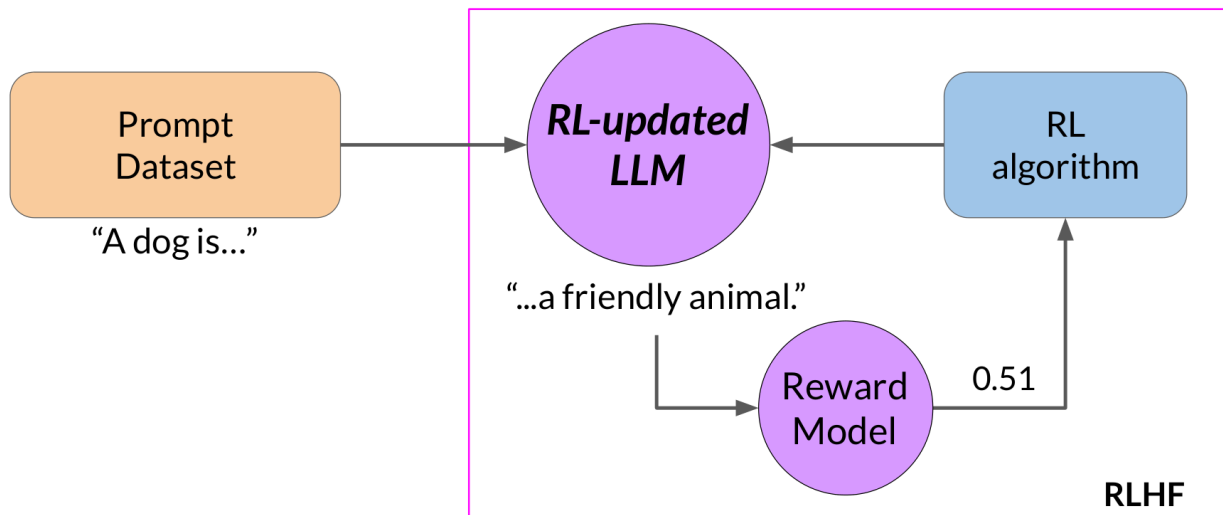


Iteration 1

Fine-tuning with RLHF

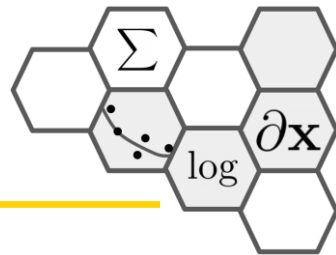


- Use the reward model to fine-tune LLM with RL

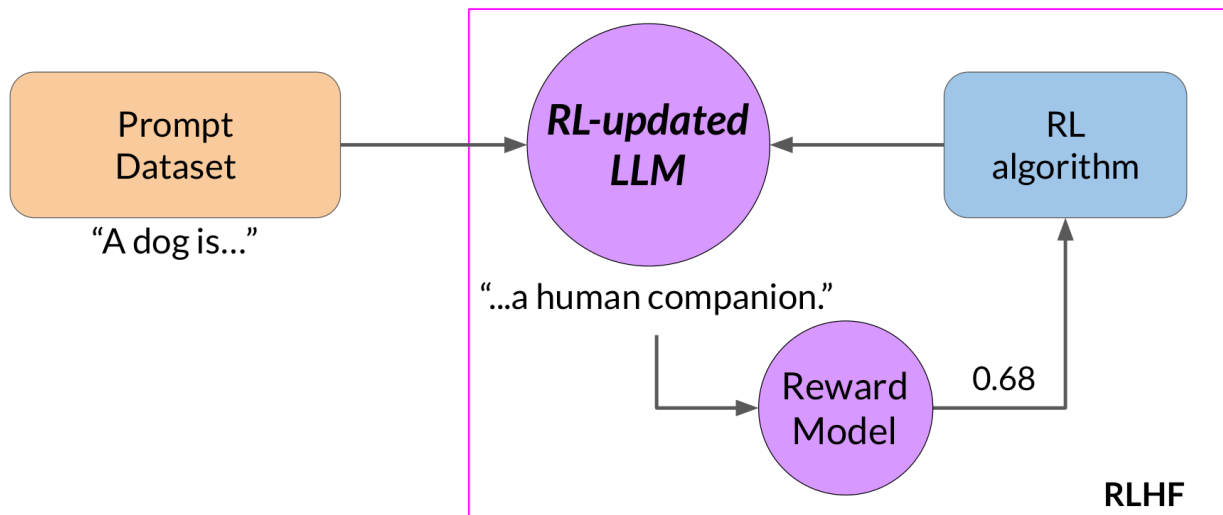


Iteration 2

Fine-tuning with RLHF

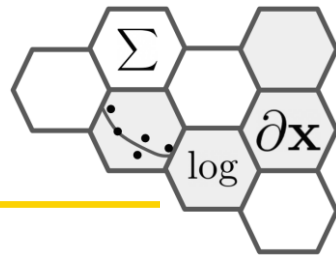


- Use the reward model to fine-tune LLM with RL

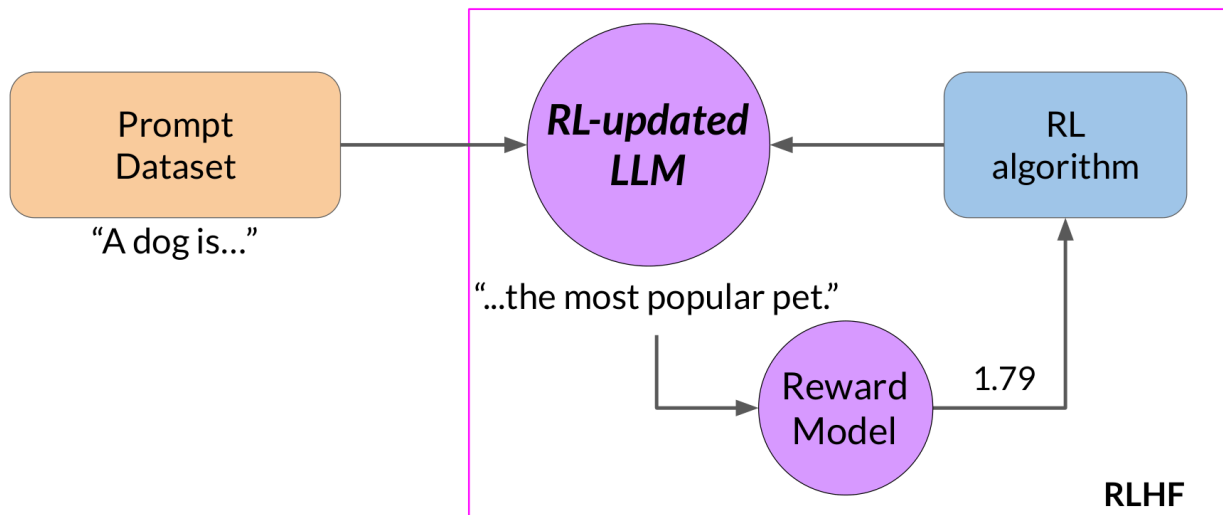


Iteration 3

Fine-tuning with RLHF

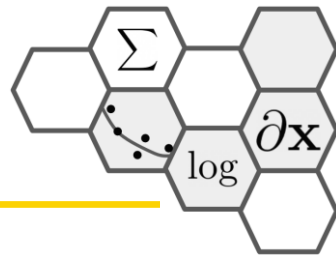


- Use the reward model to fine-tune LLM with RL

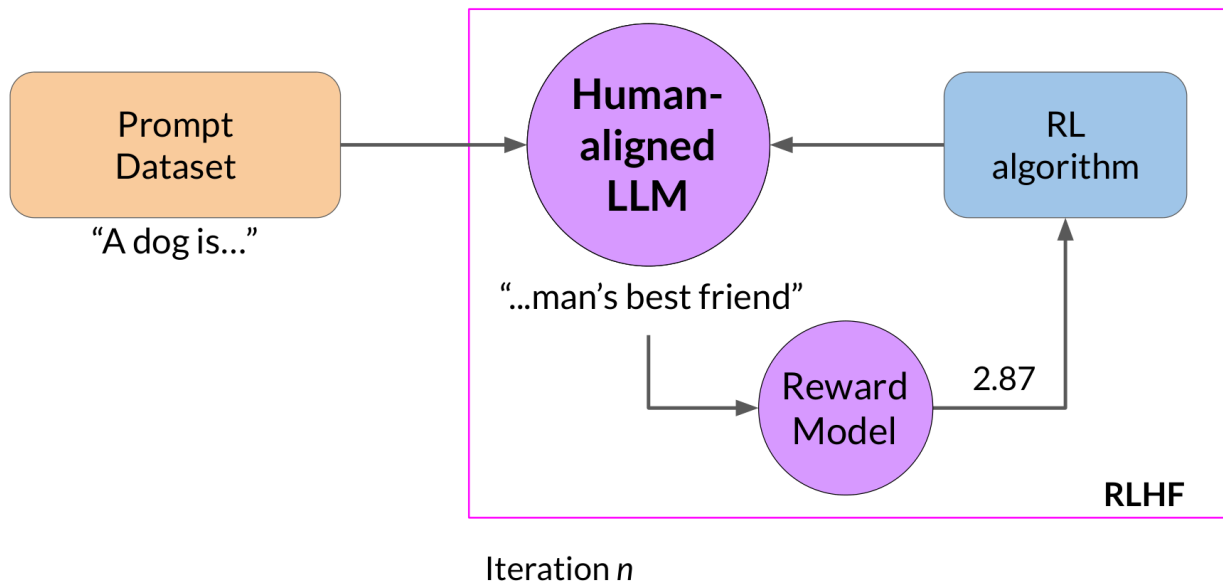


Iteration 4...

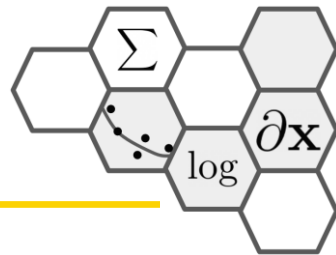
Fine-tuning with RLHF



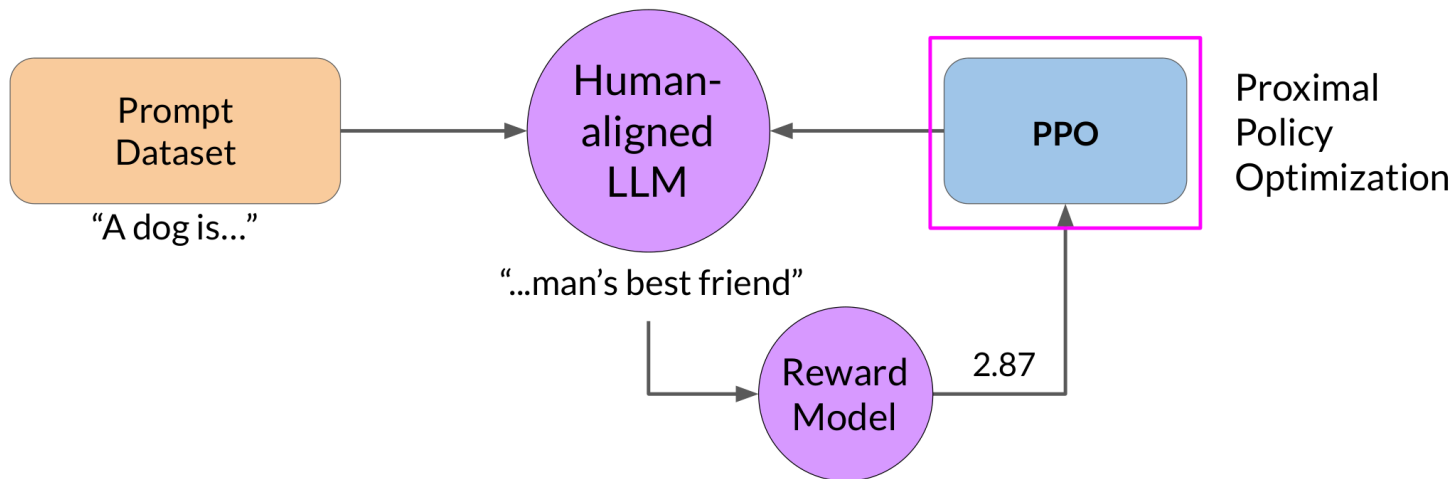
- Use the reward model to fine-tune LLM with RL



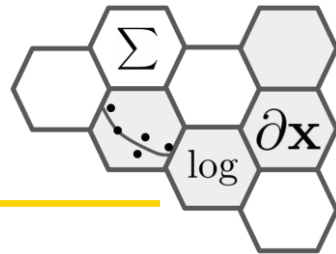
Fine-tuning with RLHF



- Use the reward model to fine-tune LLM with RL

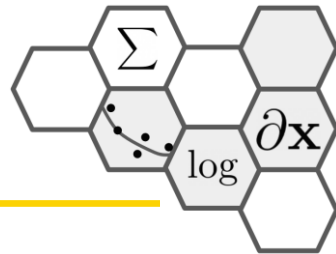


RAG

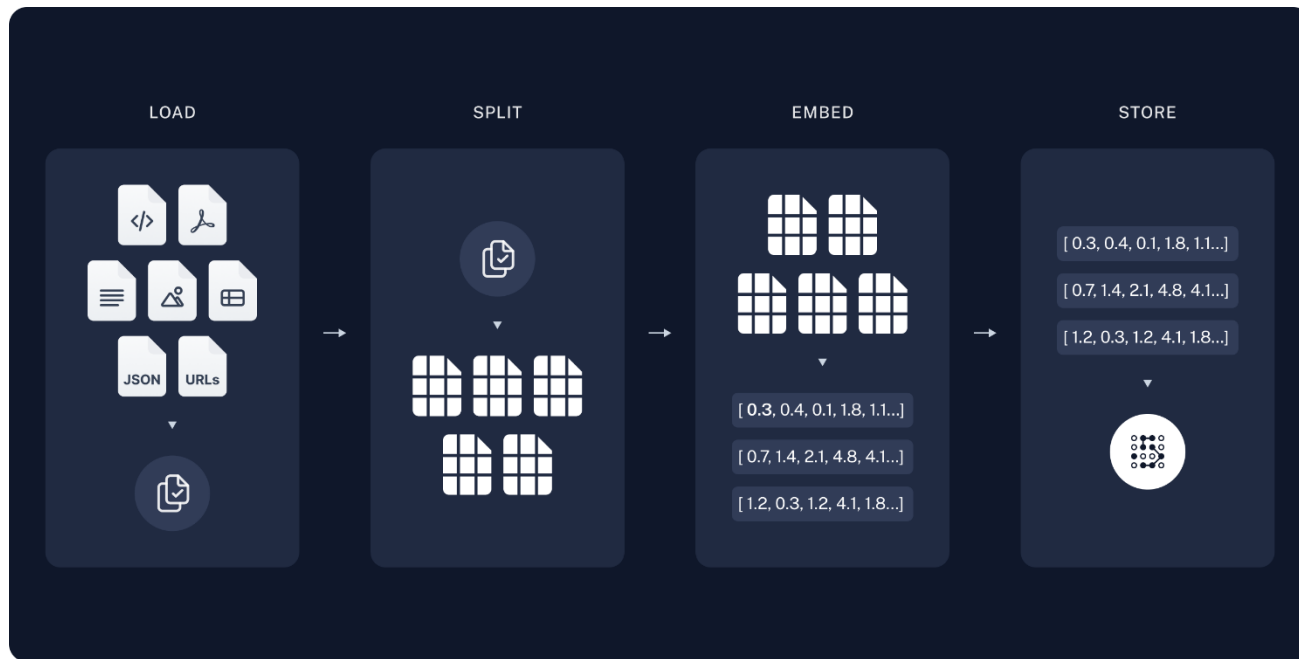


- RAG(Retrieval-Augmented Generation)?
 - LLM의 생성 능력과 외부 지식을 결합하는 기술
 - 실시간으로 관련 정보를 검색하여 응답 생성
 - 최신 정보를 반영한 정확한 답변 가능
- RAG의 작동 원리
 - 질문/프롬프트 입력 받음 \Rightarrow 관련 문서/데이터 검색 \Rightarrow 검색된 정보를 컨텍스트로 활용 \Rightarrow LLM이 최종 응답 생성
- 주요 구성 요소
 - 검색 시스템
 - 벡터 데이터베이스: [FAISS](#), Chroma, Pinecone
 - 임베딩 모델: Sentence Transformer, Claude Voyage, [OpenAI text-embedding-3](#)
 - 생성 모델(LLM): [ChatGPT](#), Claude 등
 - 통합 파이프라인: [LangChain](#), LlamaIndex
- 목적
 - 비공개 데이터나 모델의 마감일 이후에 도입된 데이터에 대해 추론할 수 있는 AI 애플리케이션을 구축

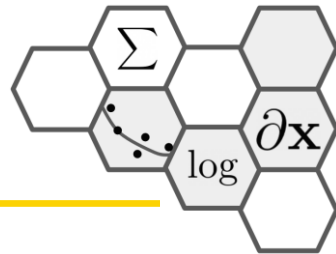
RAG



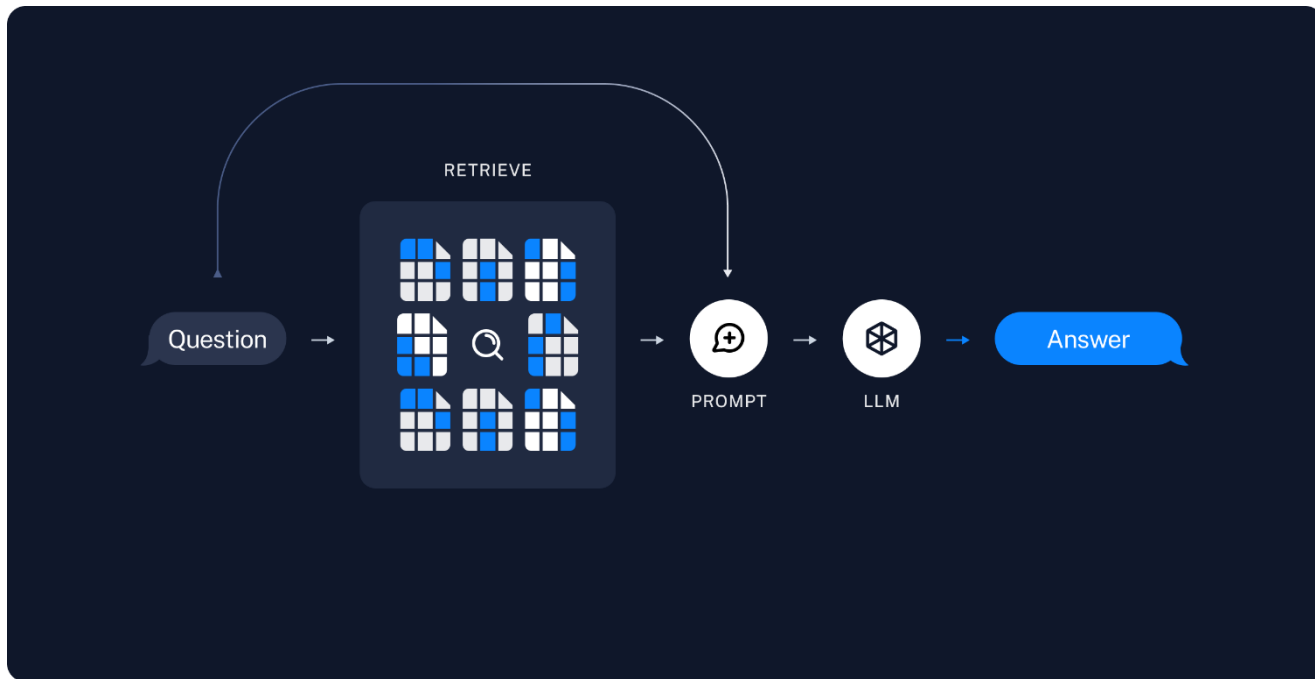
- Indexing



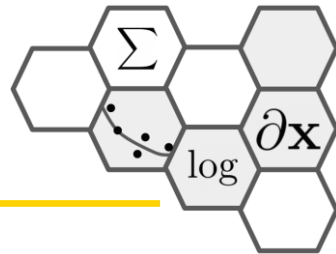
RAG



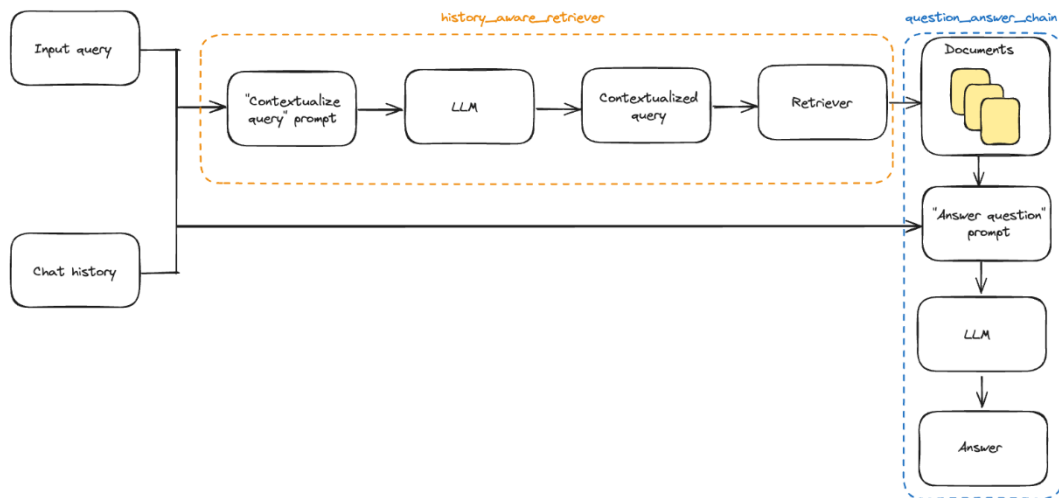
- Retrieval and generation



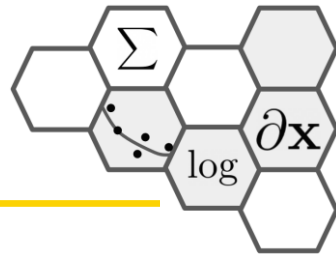
RAG



- Official Tutorials
 - Retrieval Augmented Generation (RAG): Build an application that uses your own documents to inform its responses.
 - Conversational RAG: Build a RAG application that incorporates a memory of its user interactions.



RAG



- Conversational RAG

- Build a RAG application that incorporates a memory of its user interactions.
- Tool: streamlit + LangChain + OpenAI API

문무대왕함은 몇톤급인가요?

문무대왕함은 3,600톤급의 잠수함입니다. 추가적인 정보가 필요하시면 말씀해 주세요!

광개토 대왕 급인가요?

네, 문무대왕함은 광개토대왕급 잠수함입니다. 더 궁금한 점이 있으면 말씀해 주세요!

이지스함이라고 할 수 있나요?

문무대왕함은 이지스 시스템을 탑재한 함정이 아닙니다. 이지스함은 주로 이지스 전투 시스템을 사용하는 구축함을 의미합니다. 문무대왕함은 잠수함으로, 이지스함과는 다른 종류의 함정입니다.

without RAG

문무대왕함은 몇톤급인가요?

문무대왕함은 4천 톤 급의 구축함입니다.

광개토 대왕 급인가요?

문무대왕함은 광개토대왕급이 아닌 4천 톤 급의 충무공 이순신 급 구축함입니다.

이지스함이라고 할 수 있나요?

현재 문무대왕함은 이지스 체계를 갖추고 있지 않지만, 머지않아 7천 톤 급의 구축함 시대에 이지스 체계를 갖춘 함정이 실현될 예정입니다. 따라서 문무대왕함 자체는 이지스함이라고 할 수 없습니다.

with RAG