Procesamiento de Lenguaje Natural Avanzado

De transformers a los LLMs

Dra. Helena Gómez Adorno
helena.gomez@iimas.unam.mx

Dr. Fazlourrahman Balouchzahi
fbalouc@iimas.unam.mx

Correo del curso:
pln.cienciadedatos@gmail.com

Dos formas de usar los modelos de lenguaje

Ajuste fino

- Descenso de gradiente en pesos para optimizar el rendimiento en una tarea.

¿Qué ajustar?

- Red completa
- Cabezales de lectura
- Adaptadores

Prompting

- Diseñar una indicación especial para indicar/condicionar la red en un modo específico para resolver cualquier tarea.
- Sin cambios de parámetros. Un solo modelo para controlarlos a todos.

Cambiar el modelo mismo

Cambiar la forma en como se usar el modelo

Hungyi-Lee: https://www.youtube.com/watch?v=F58vJcGgjt0&t=4s

Ajuste fino (finetuning) vs prompting

Ajuste fino

- Se popularizó con BERT
- Proporciona representación para resolver tareas.
- Menor escala
- No puede "hablar"

Prompting

- Popularizado en GPT3/4
- Puede hablar.
- Se le puede pedir ayuda para resolver tareas.

Cambiar el modelo mismo

Cambiar la forma en como se usar el modelo

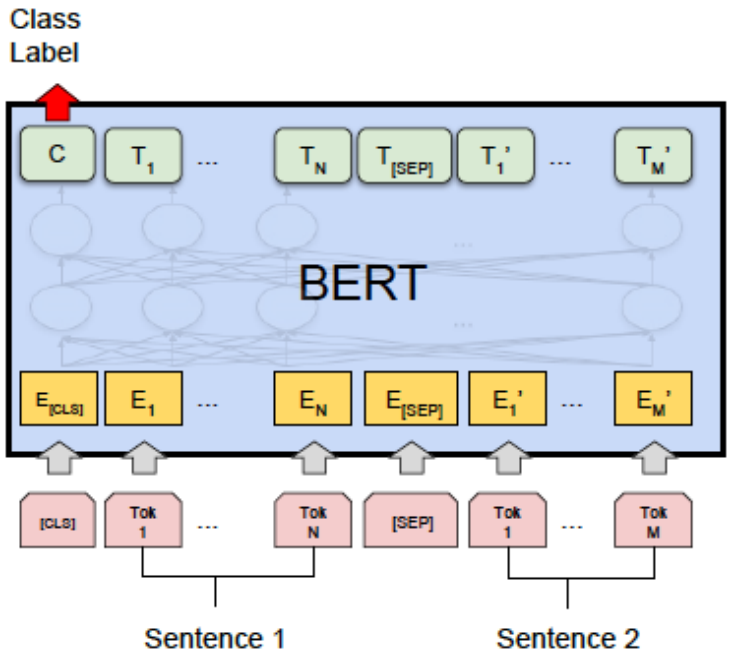
Hungyi-Lee: https://www.youtube.com/watch?v=F58vJcGgjt0&t=4s

Trajes de Ironman “Ajustados”

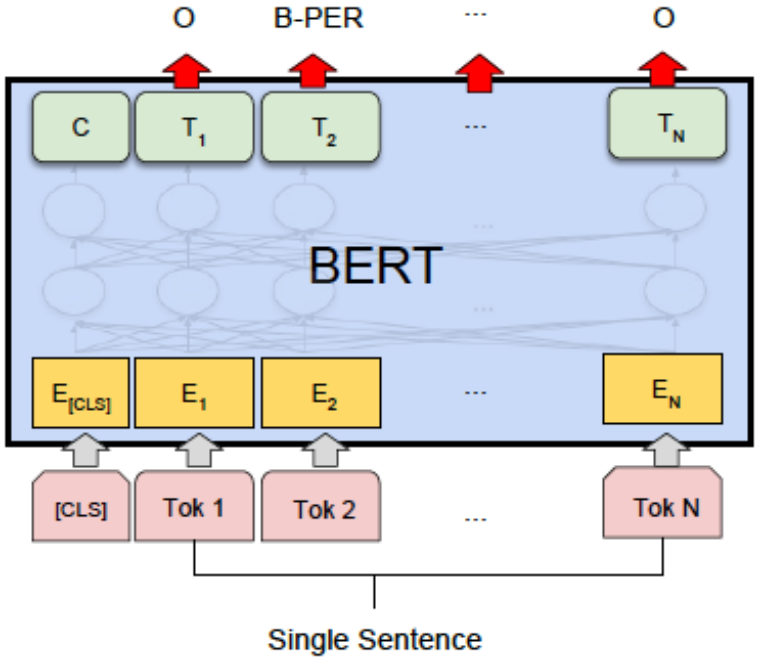
Modelo Grande Fundacional

Modelos ajustados para propósitos especiales

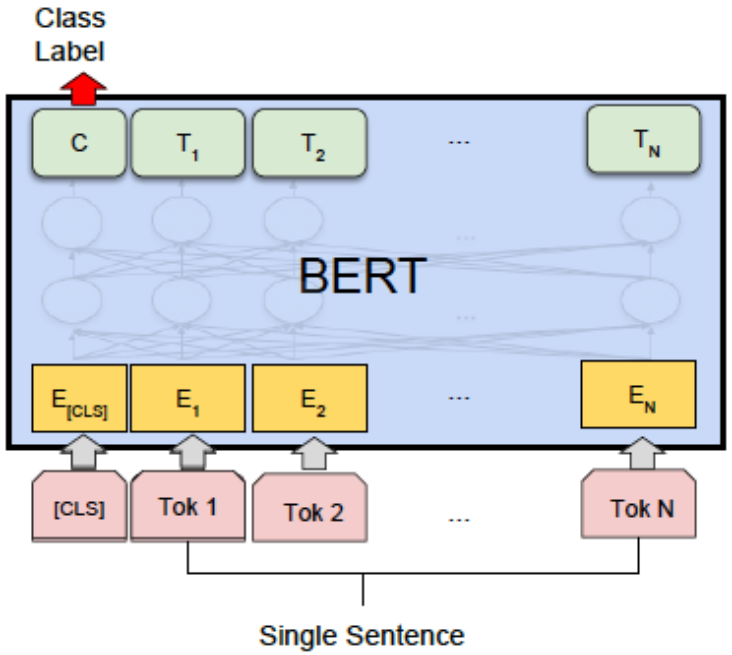
Ajuste Fino – Ejemplo de BERT



Clasificación de un par de oraciones



Etiquetado de secuencias



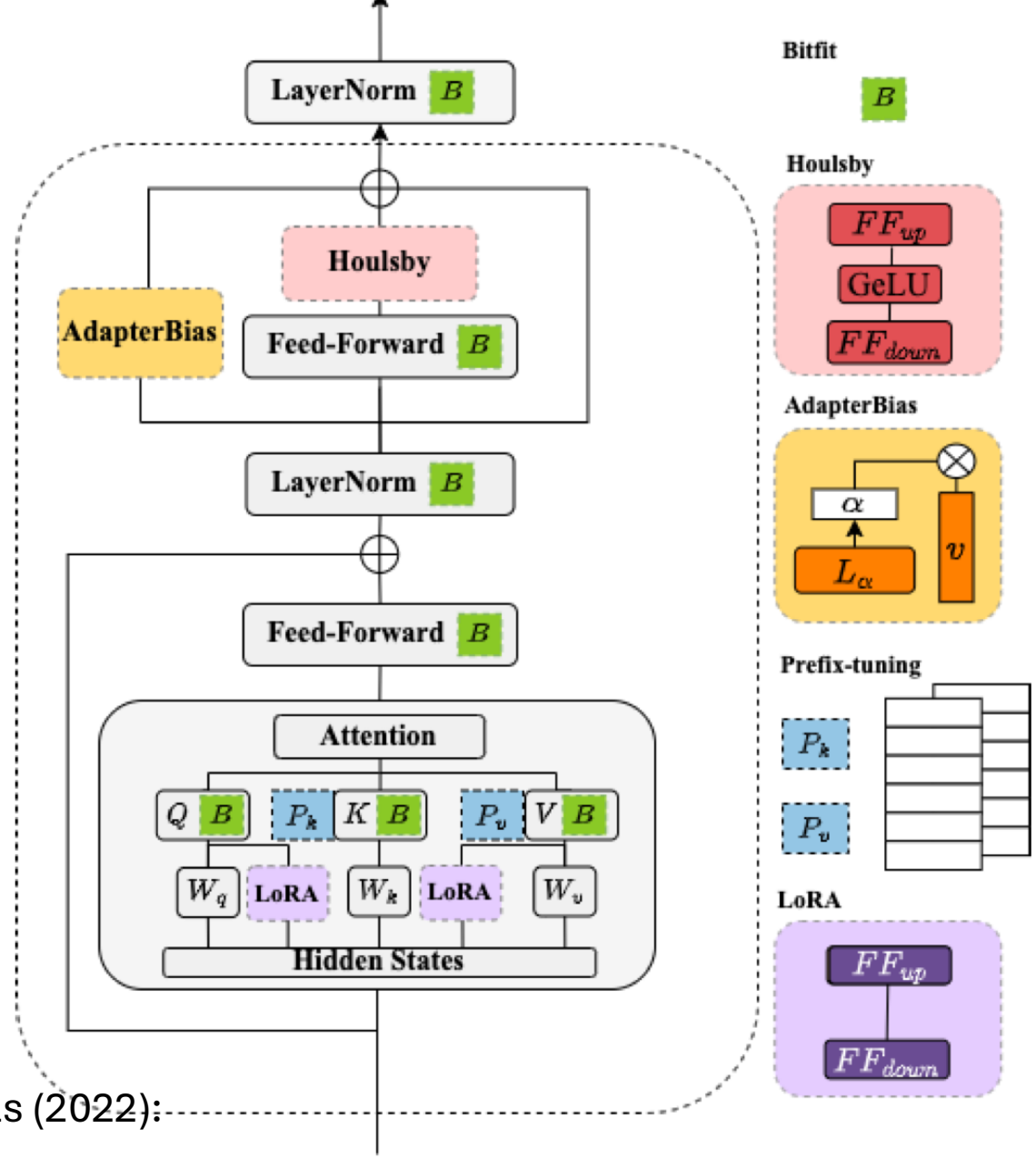
Clasificación de una oración

BERT: https://www.aclweb.org/anthology/N19-1423/

Adaptador de ajuste fino

Complemento ligero para LLM

- Inserta módulos pequeños en LLM y ajusta únicamente esos parámetros.
- Los bloques con líneas discontinuas de borde son los parámetros añadidos por el método eficiente. →
- Se guarda con mayor facilidad.



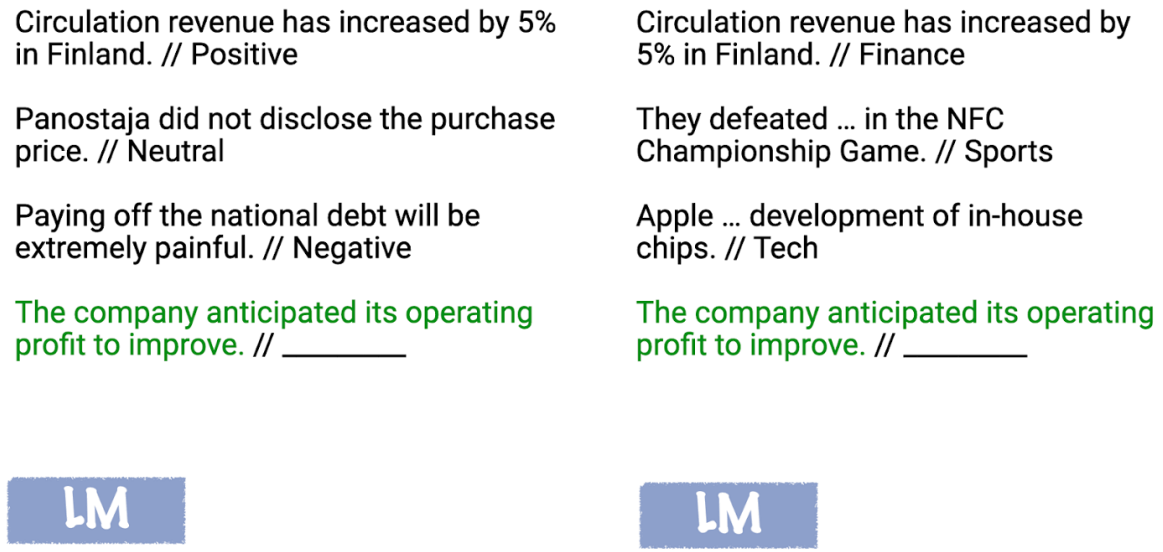
Exploring efficient-tuning methods in self-supervised speech models (2022): https://arxiv.org/pdf/2210.06175

Métodos de Adaptador – documentación: https://docs.adapterhub.ml/methods.html

Peft en Huggingface: https://huggingface.co/docs/peft/index

Prompting: dejamos que el modelo aprenda una tarea en contexto.

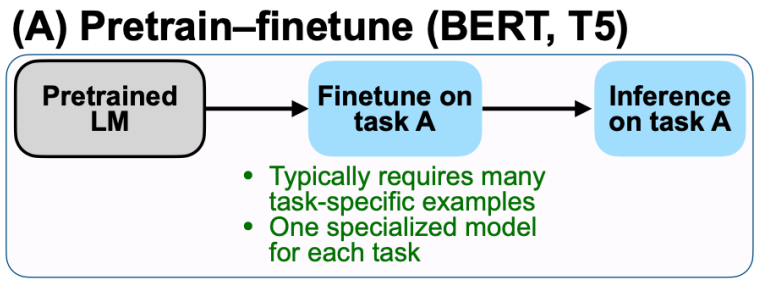
Damos algunos ejemplos como demostración y dejamos que el modelo los siga para resolver nuevos problemas.

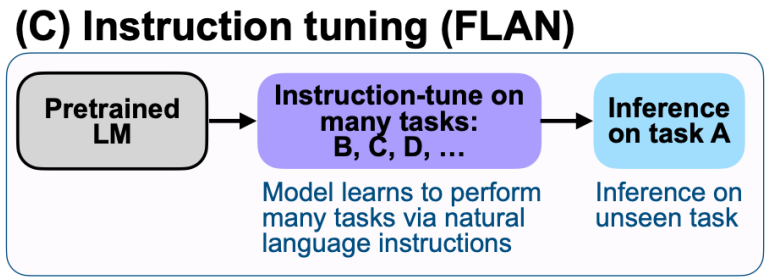


Aprendizaje en contexto (In context learning) - 2022: https://ai.stanford.edu/blog/understanding-incontext/

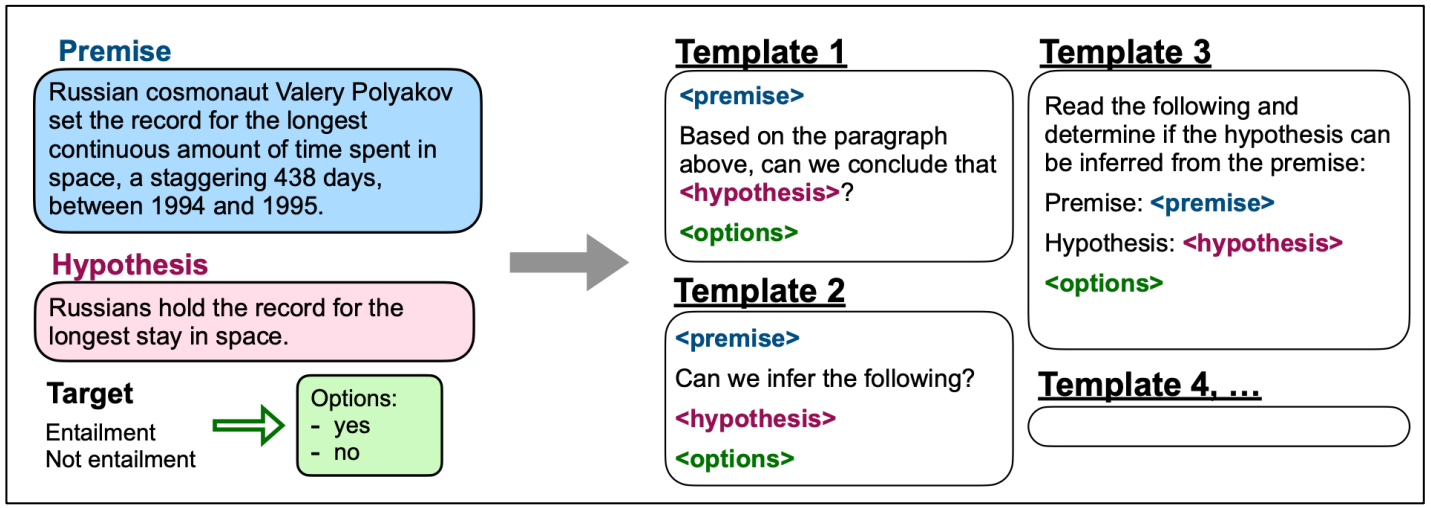
Finetuning + Prompt: Instruction Tuning

Damos algunos ejemplos como demostración y dejamos que el modelo los siga para resolver nuevos problemas.





Comparación del ajuste de instrucciones con el ajuste fino y el prompting.



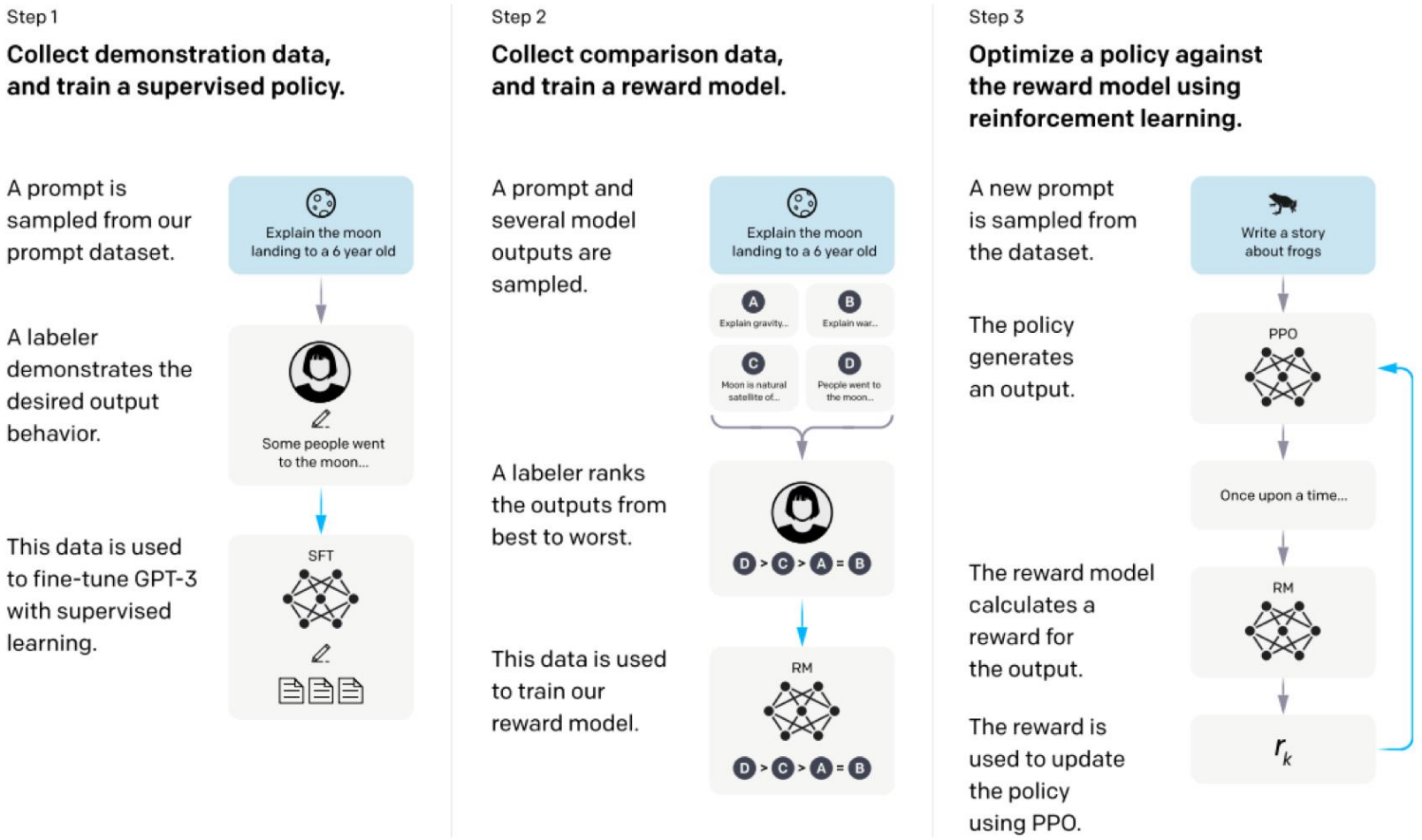
Plantillas de instrucciones que describen una tarea de inferencia.

FINETUNED LANGUAGE MODELS ARE ZERO-SHOT LEARNERS (2021): https://arxiv.org/pdf/2109.01652

Instruct GPT / RLFH

Método de entrenamiento detrás de ChatGPT

- Ajustaron GPT-3 mediante aprendizaje supervisado con un conjunto de demostraciones del comportamiento deseado del modelo.
- Luego ajustaron aún más este modelo supervisado mediante el aprendizaje de refuerzo a partir de la retroalimentación humana.



Supervised learning
Similar to FLAN

Reinforcement learning from human feedback

InstructGPT https://arxiv.org/abs/2203.02155

Why align?

Pretraining

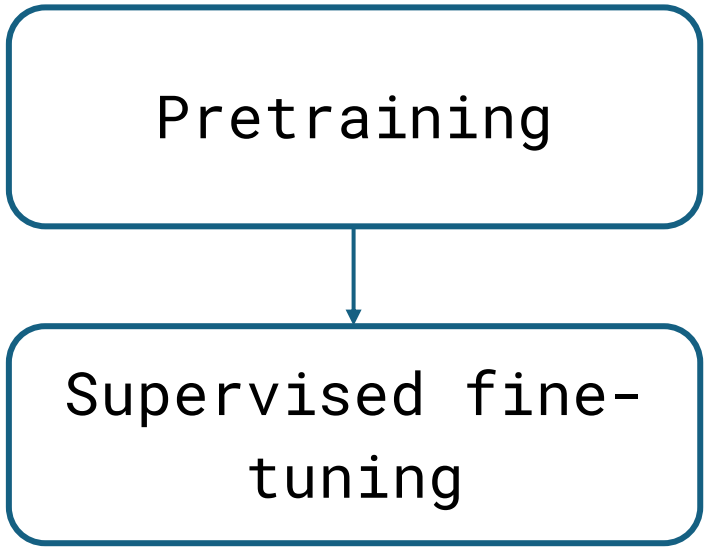
Is pineapple on pizza a crime?

Base LLM

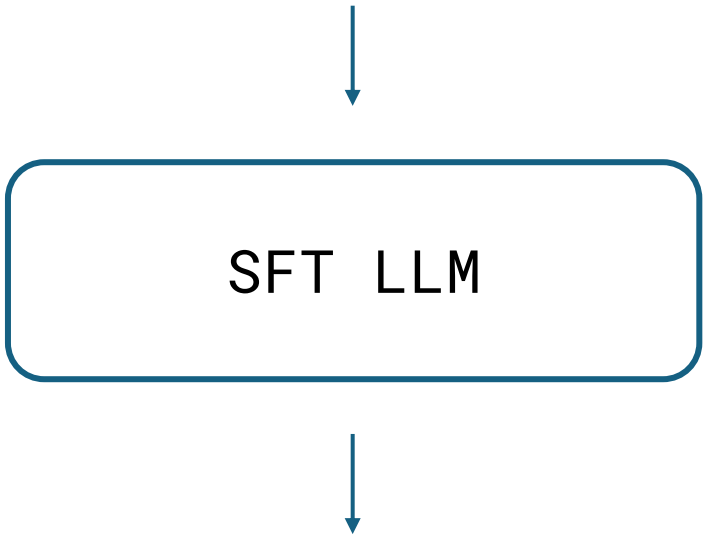
This is one of the many questions that will be answered at the Pizza Party ...

Diapositivas tomadas de Hugging face

Why align?

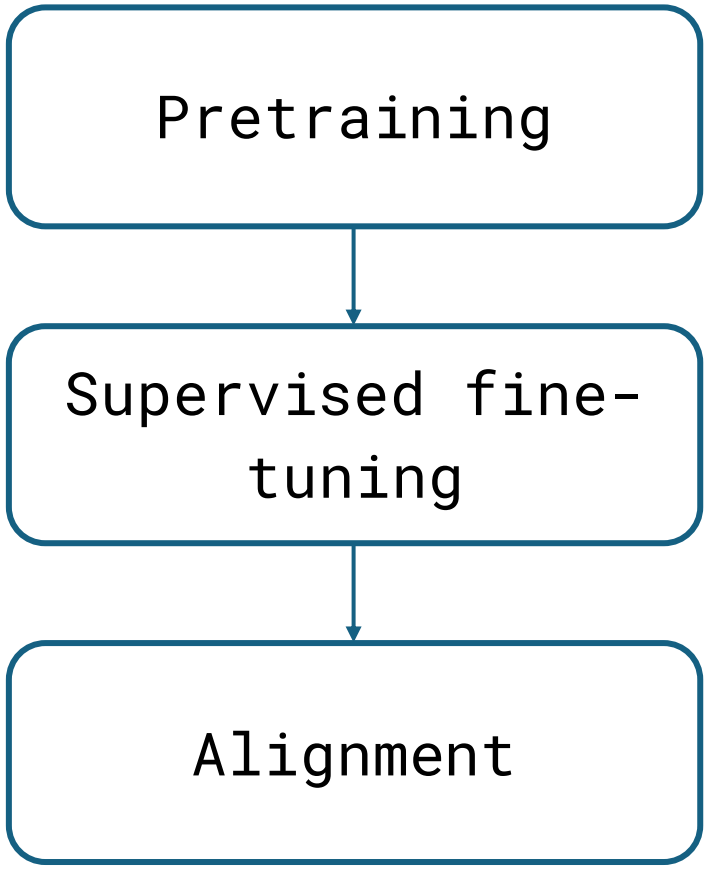


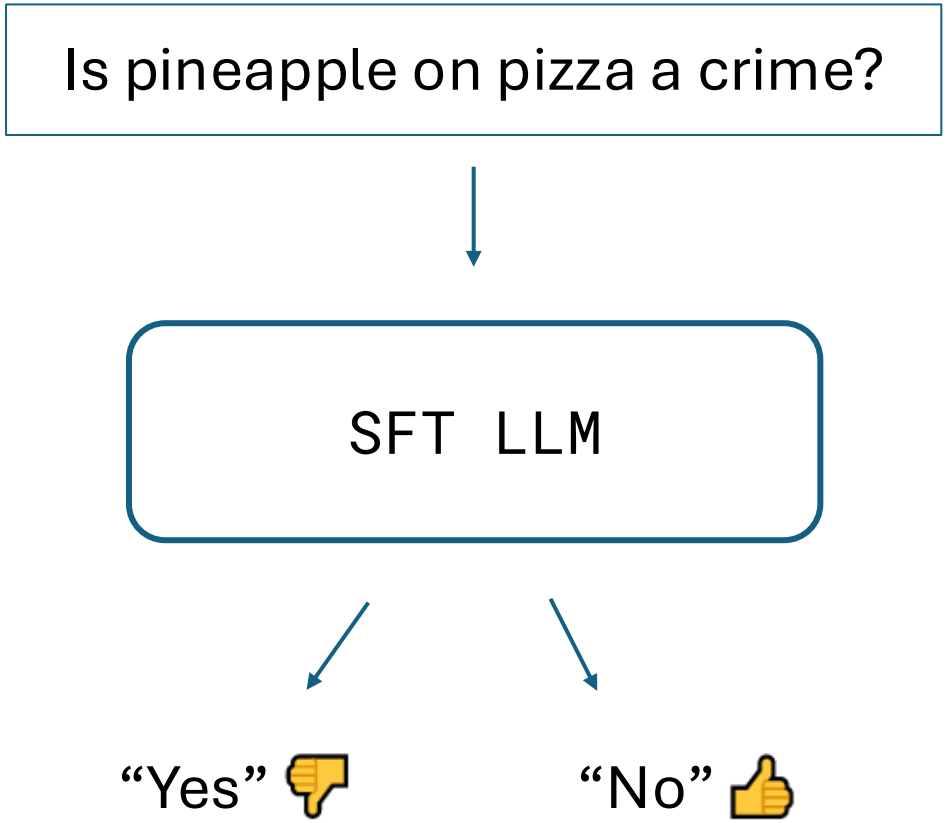
Is pineapple on pizza a crime?



Yes, adding pineapple as a topping on pizza is a criminal act under the Geneva Convention

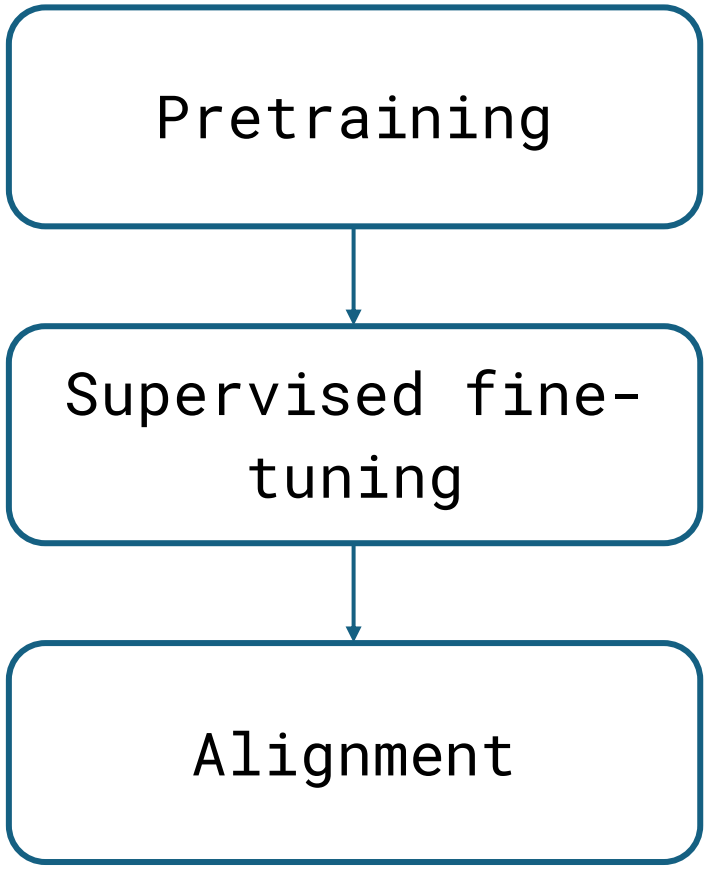
Why align?



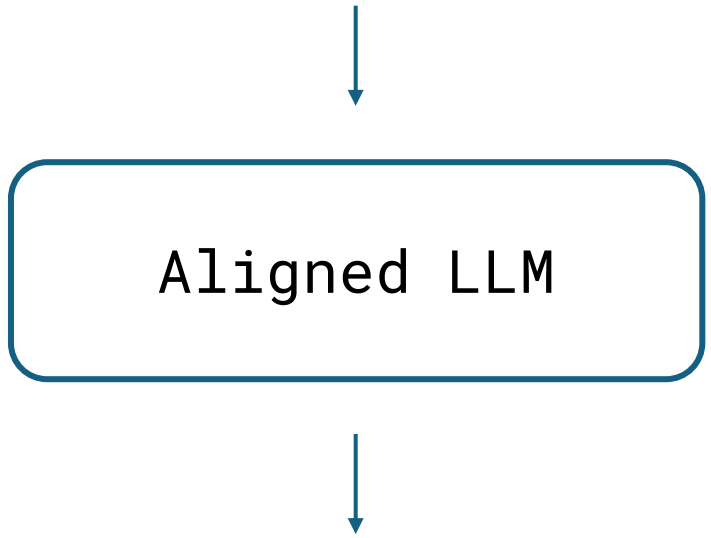


Collect human / AI feedback
to learn $p(y_w \succ y_l)$

Why align?



Is pineapple on pizza a crime?



No, adding pineapple as a topping on pizza is not a criminal act. It's a matter of personal preference and taste.

RLHF- the begining of LLM alignment

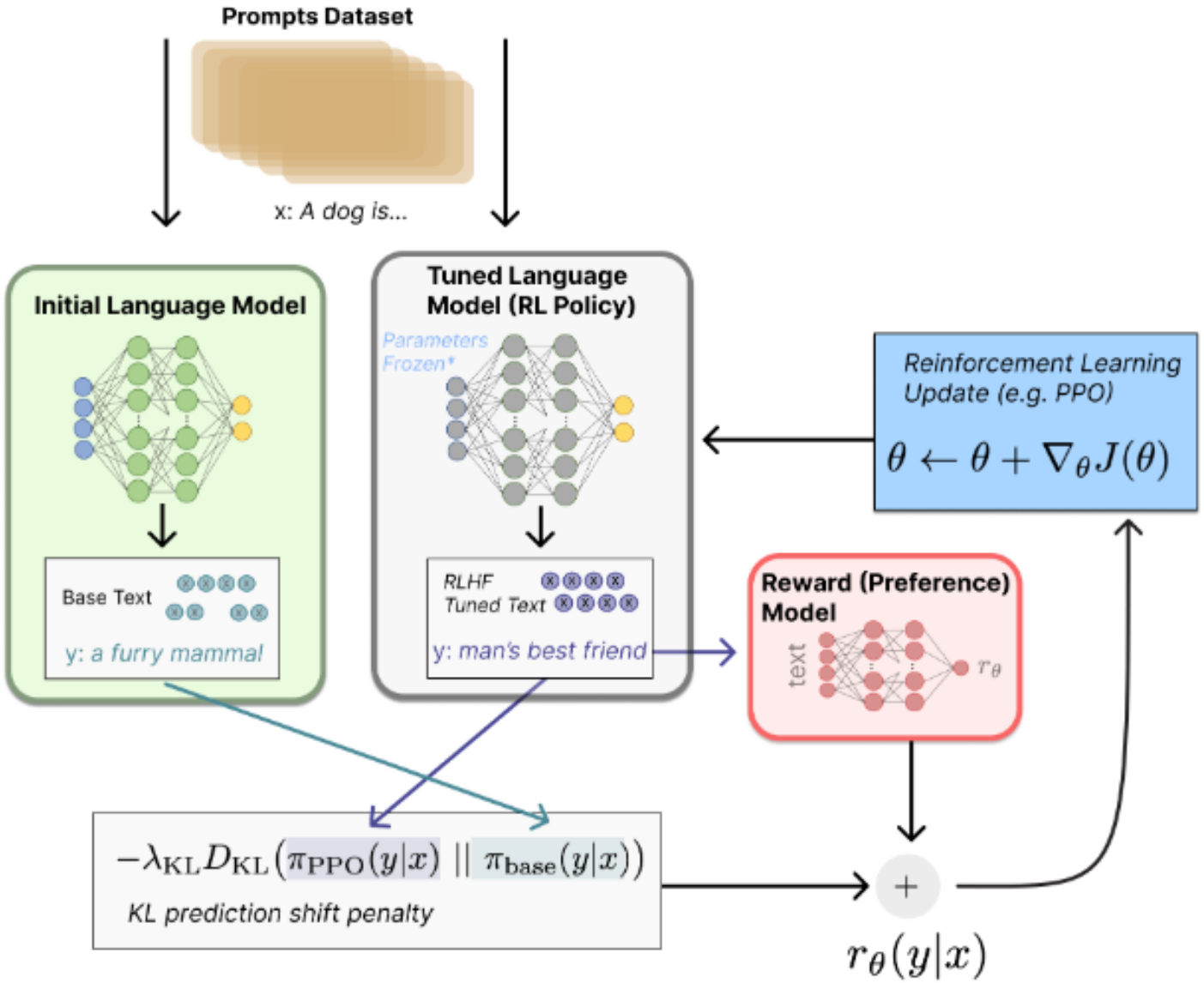
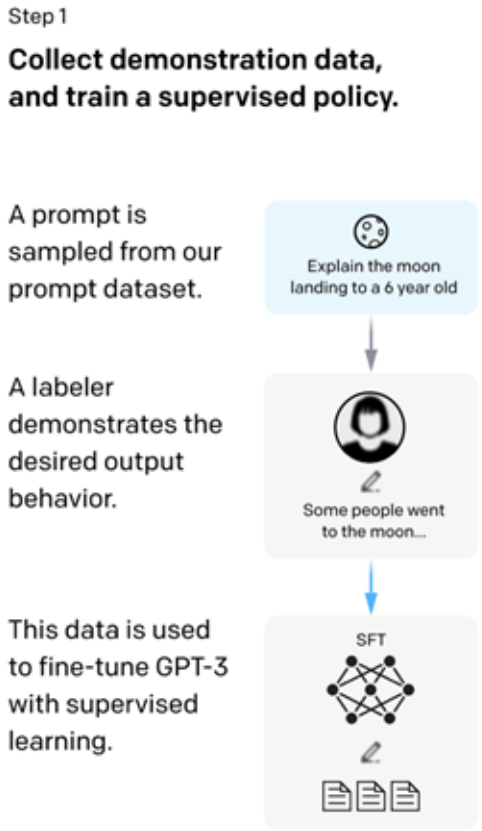


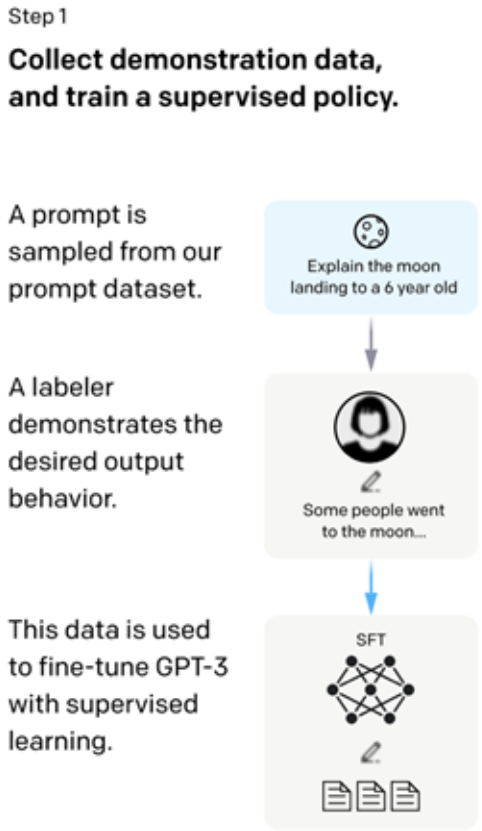
Imagen de: https://huggingface.co/blog/rlhf

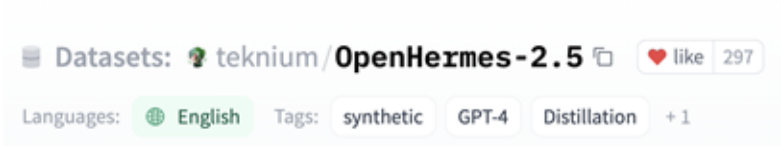
RLHF- the begining of LLM alignment

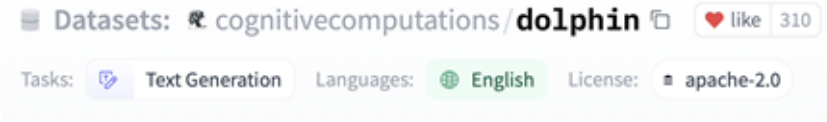


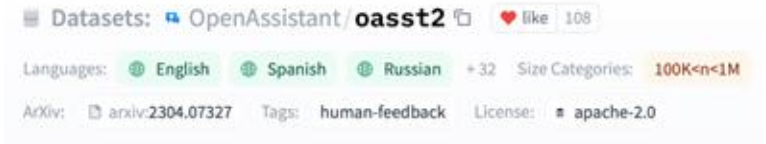
Stiennon, Nisan, et al. "Learning to summarize with human feedback." Advances in Neural Information Processing Systems 33 (**2020**): 3008-3021.
Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Lowe, R. (**2022**). Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, *35*, 27730-27744.

RLHF- the begining of LLM alignment



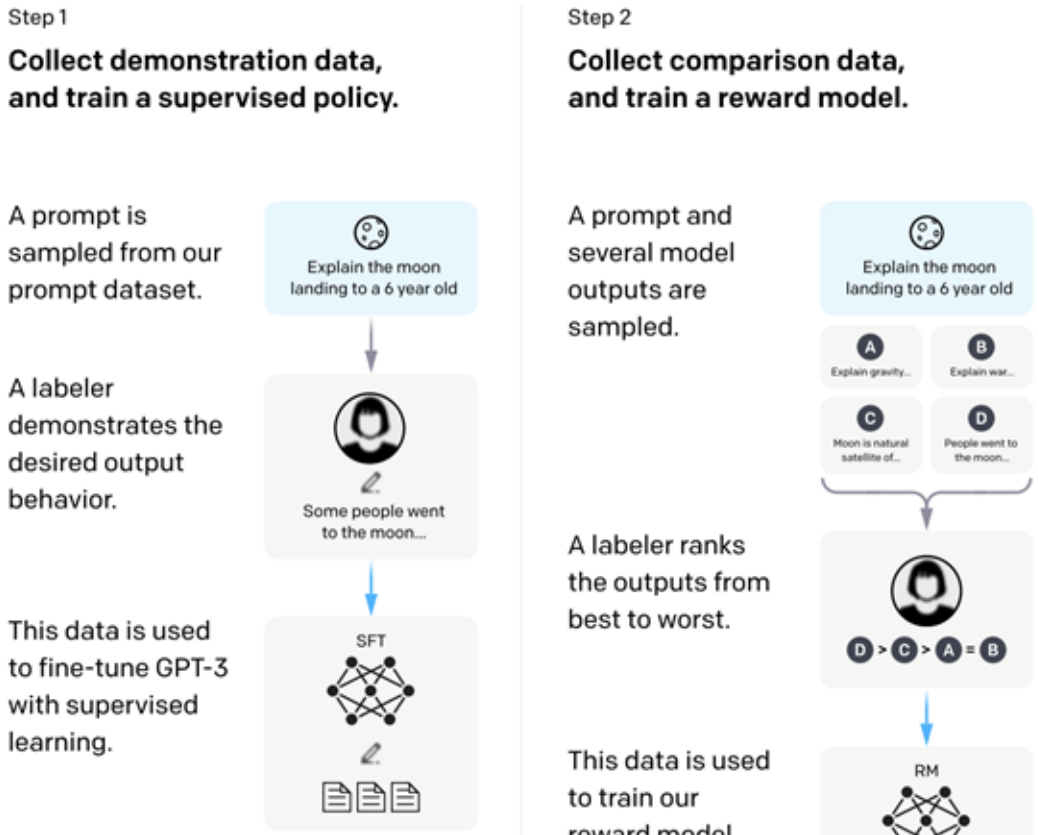






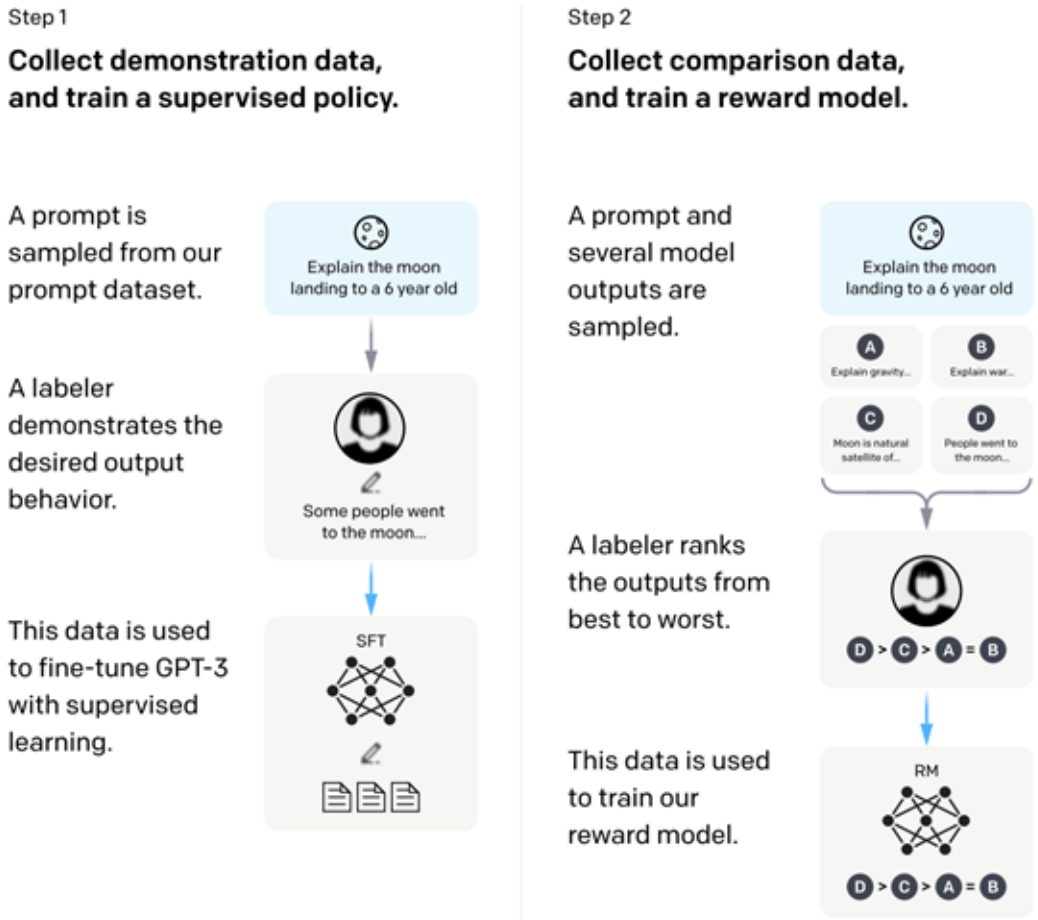
Stiennon et al (2020)
Ouyang et al (2022)

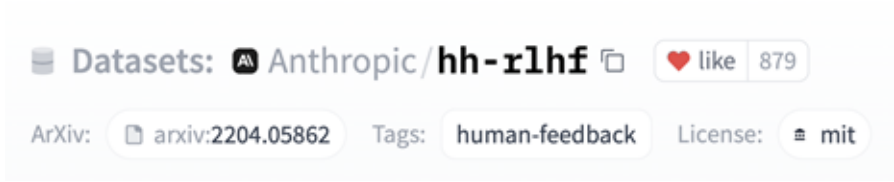
RLHF- the begining of LLM alignment

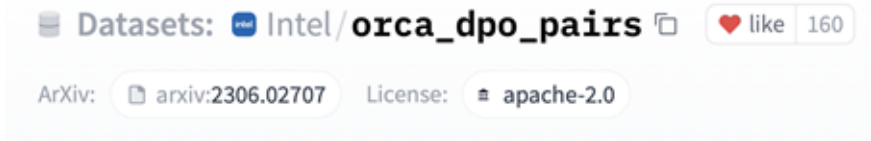


Stiennon et al (2020)
Ouyang et al (2022)

RLHF- the begining of LLM alignment

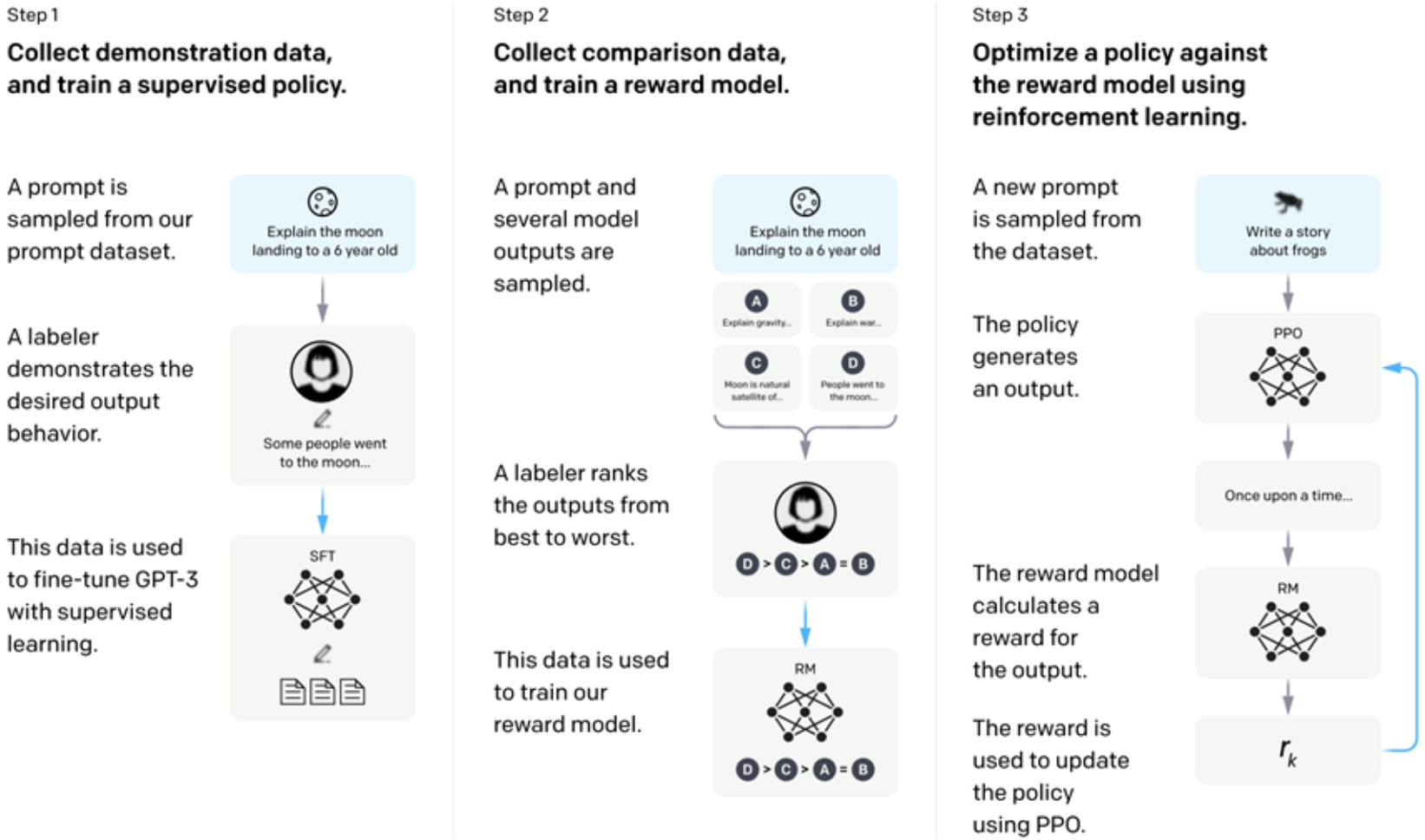






Stiennon et al (2020)
Ouyang et al (2022)

RLHF- the begining of LLM alignment



Stiennon et al (2020)
Ouyang et al (2022)

RLHF- the begining of LLM alignment

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r_\phi(x, y) \right] - \beta \mathbb{D}_{\mathrm{KL}} \left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x) \right]$$

maximise rewards

use KL-divergence penalty to prevent **reward hacking** (controlled by β)

Varios Retos

RL notoriamente inestable, muchos hiperparametros
Necesita modelo de recomensa separado
⇒ 3 LLMs para compaginar 😱


The N Implementation Details of RLHF with PPO

Published October 24, 2023

Update on GitHub

vwxyzjn Shengyi Costa Huang
tianlinliu0121 Tianlin Liu guest
lvwerra Leandro von Werra

https://huggingface.co/blog/the_n_implementation_details_of_rlhf_with_ppo

Direct Preference Optimization

$$\max_{\pi} \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma \left(\beta \log \frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." *Advances in Neural Information Processing Systems* 36 (2023): 53728-53741.

Direct Preference Optimization

$$\max_{\pi} \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}$$

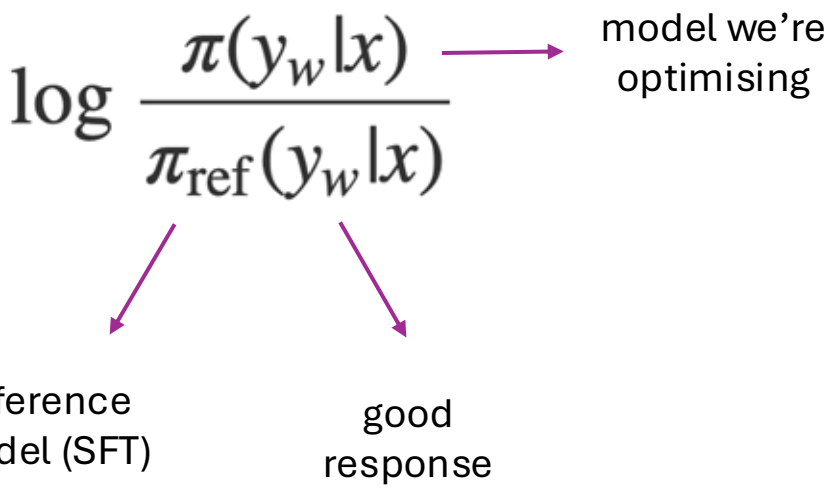
good response

bad response

(Is pineapple on pizza a crime?, No, Yes)

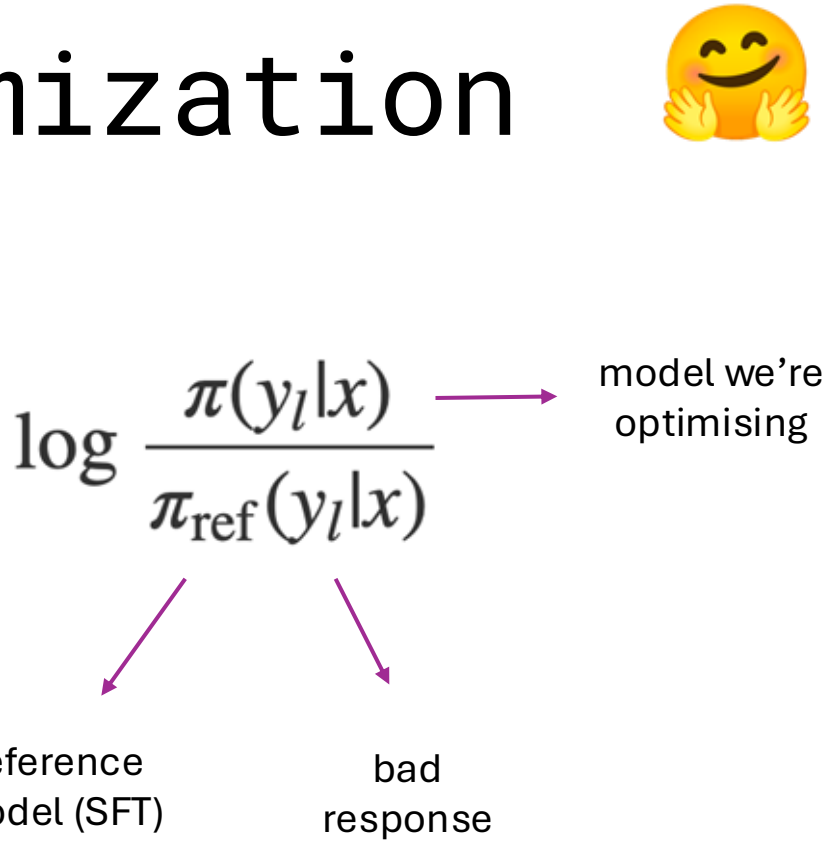
Rafailov et al (2023)
AIF + DPO: Distilling Zephyr and friends

Direct Preference Optimization



Rafailov et al (2023)
AIF + DPO: Distilling Zephyr and friends

Direct Preference Optimization



Rafailov et al (2023)
AIF + DPO: Distilling Zephyr and friends

Direct Preference Optimization

$$\max_{\pi} \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma \left(\beta \log \frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

```python
import torch.nn.functional as F

def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta):
    pi_yw_logps,  pi_yl_logps =  pi_logps[yw_idxs],  pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]
    pi_logratios  = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps
    losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios))
    rewards = beta * (pi_logps - ref_logps).detach()
    return losses, rewards
```

Algoritmo

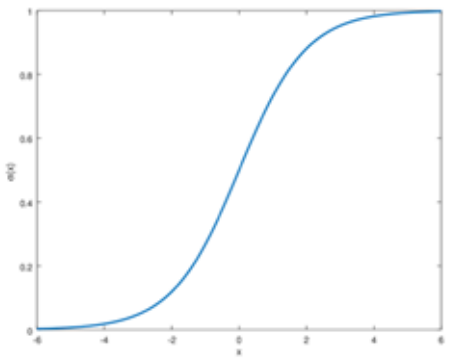
- Sample good/bad response
- Run pairs through 2 models (active and reference)
- Backprop
- Profit 💪

Rafailov et al (2023)

What does the DPO update do?

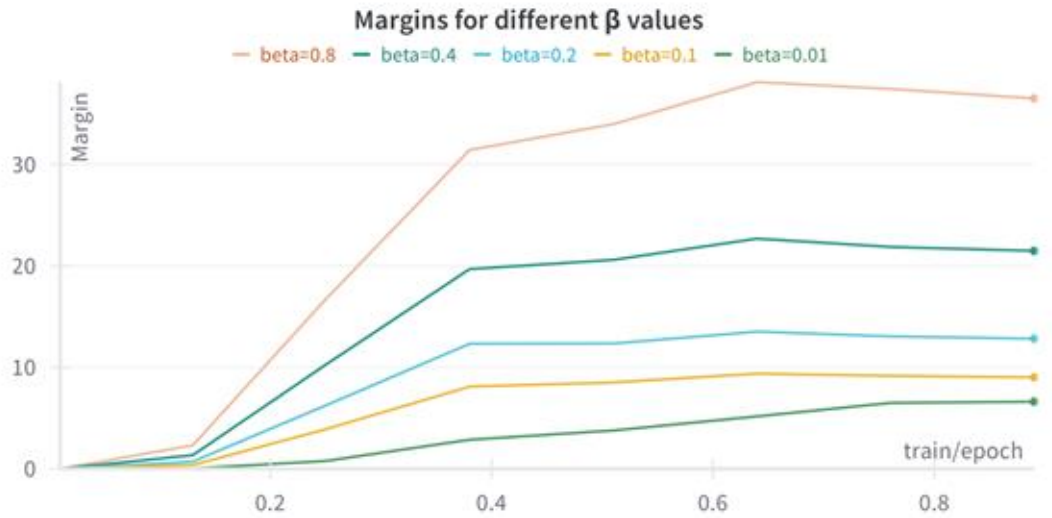
$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big],$$



$$\hat{r}_\theta(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

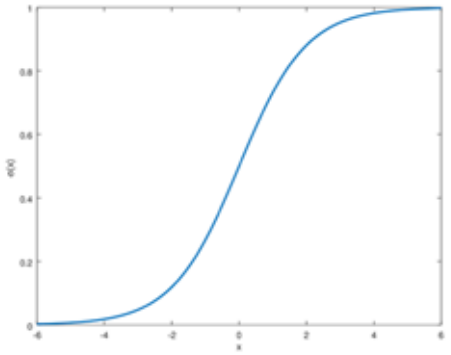
implicit reward from LM



What does the DPO update do?

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

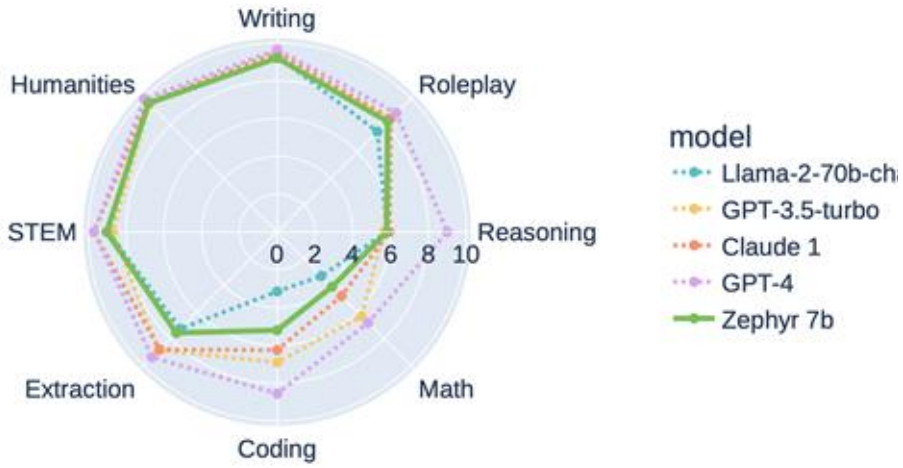
$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Big[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Big]\Big],$$



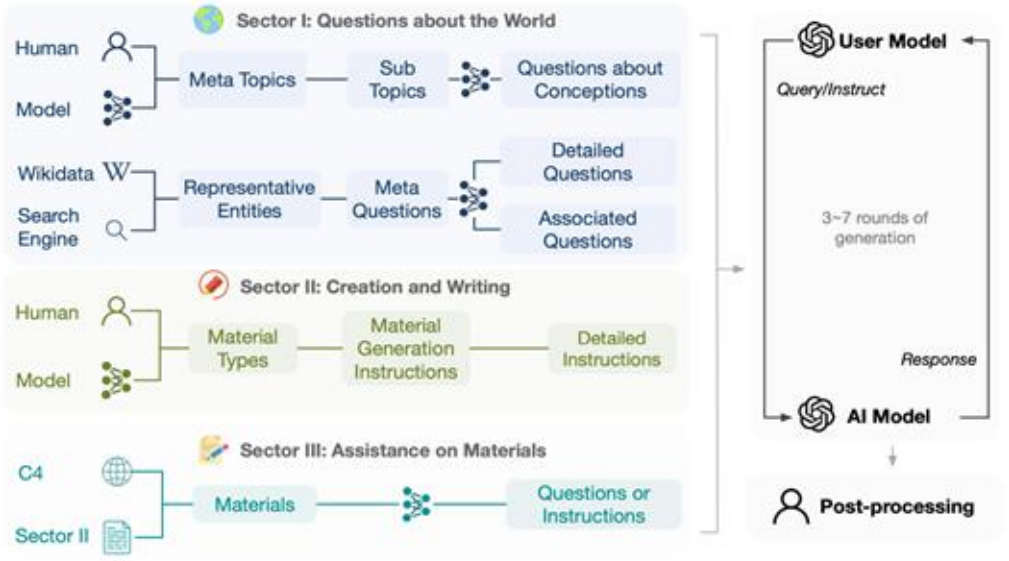
$$\hat{r}_\theta(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

implicit reward from LM

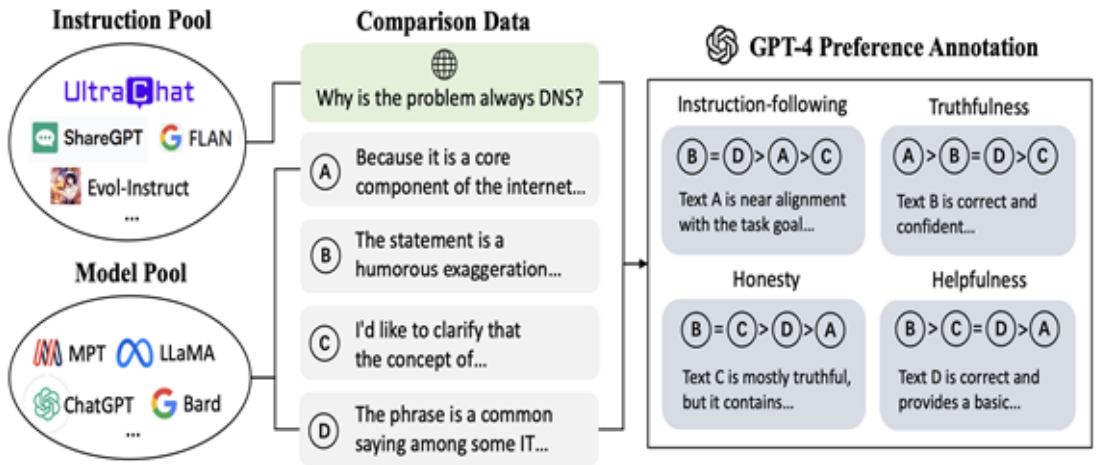
Some examples



MT Bench



UltraChat
for SFT
Ding et al (2023)



UltraFeedback
for DPO
Cui et al (2023)

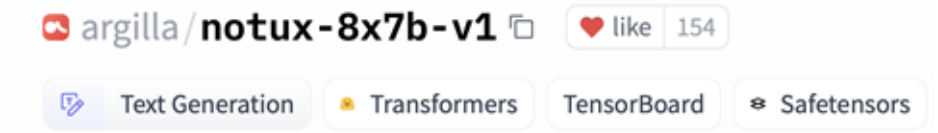
Tunstall et al (2023)

Some examples

huggingface/trl

Axolotl provides a unified repository for fine-tuning a variety of AI models with ease

OpenAccess-AI-Collective/axolotl



Going beyond DPO

DPO

IPO

Azar et al (2023)

Adds a regularisation term to prevent overfitting

KTO

Ethayarajh et al (2023)

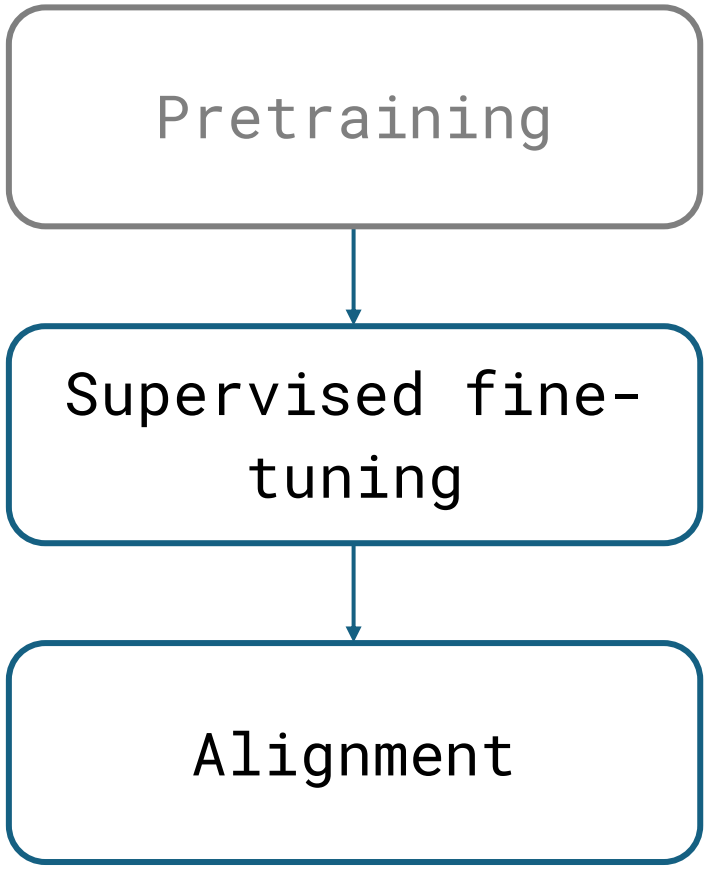
Dispenses with binary preferences altogether!

Iterative DPO

Snorkel (2023)

Combines rejection sampling with DPO

Training and Aligning a Chatbot 🤗



- Practical dive into SFT and DPO
- Introduce Chat Templates for formatting dialogue data
- Links for SFT and DPO datasets
- Metrics
- Model Evaluation

Annotated SFT & DPO

Notebooks: (on colab)

- Annotated SFT
- Annotated DPO

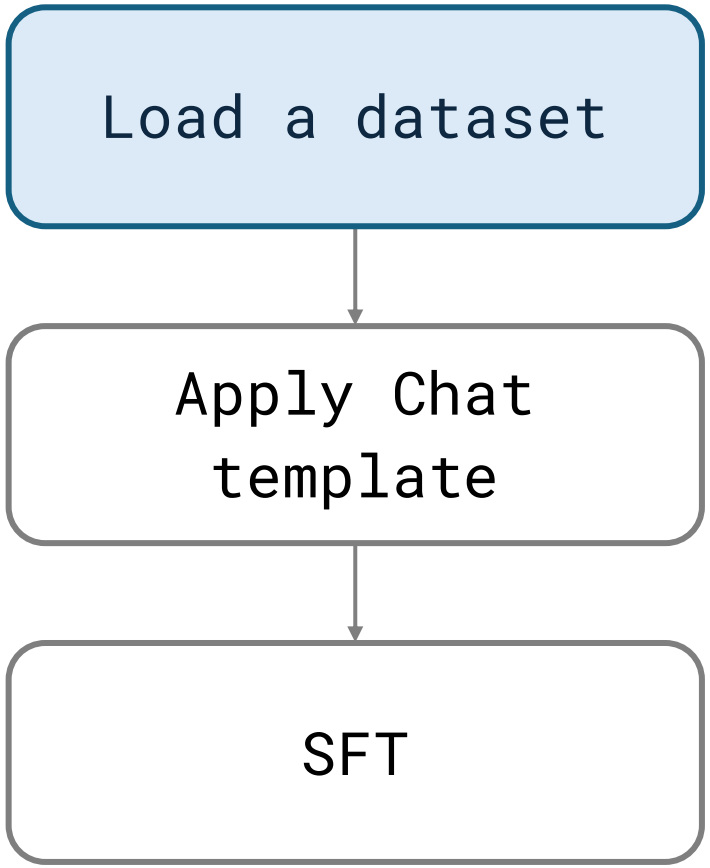
More up to date codebase: Hugging Face Alignment Handbook

- Low-resource examples with Q-LORA
- Multi-GPU / node examples with Accelerate & DeepSpeed
- Configs, hyper-parameters, slurm scripts

A note on LORA:

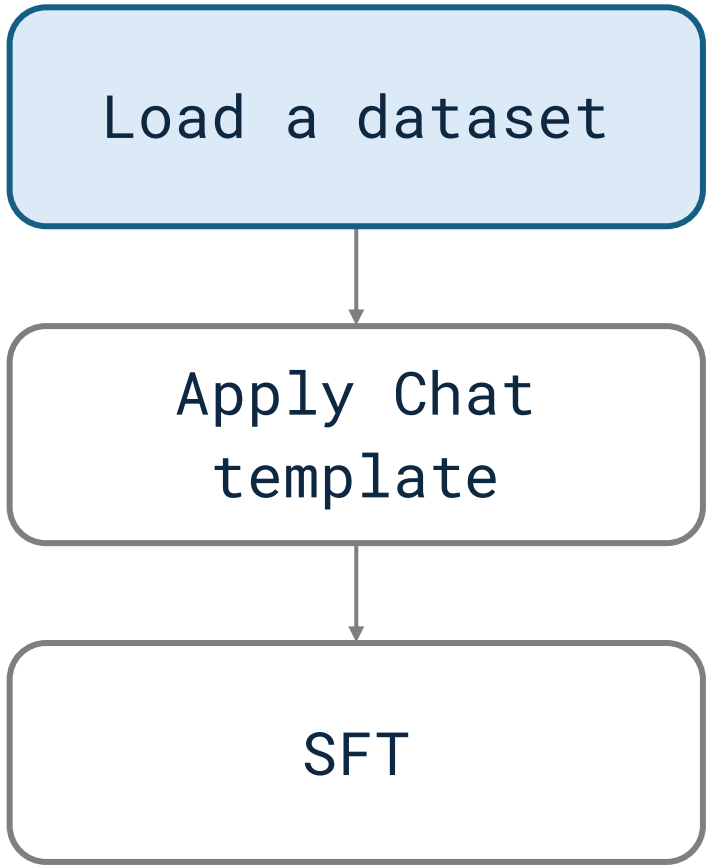
- PEFT blogpost
- LORA paper

Supervised Fine-Tuning (SFT)




```
from datasets import load_dataset

raw_datasets = load_dataset("HuggingFaceH4/ultrachat_200k")
print(raw_datasets)

DatasetDict({
      train: Dataset({
          features: ['prompt', 'prompt_id', 'messages'],
          num_rows: 200000
      })
      test: Dataset({
          features: ['prompt', 'prompt_id', 'messages'],
          num_rows: 2000
      })
})
```

Supervised Fine-Tuning (SFT)



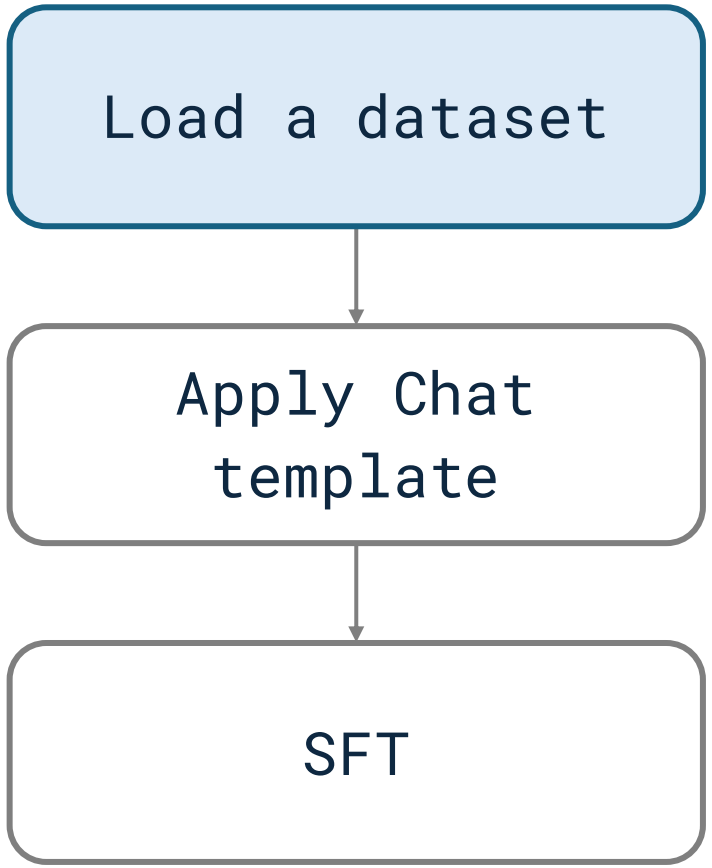
Example prompts:

Which famous landmarks should I visit in London, beyond the usual ones?

Write a program to implement a dynamic programming algorithm for the longest common substring problem in C++

Create a YouTube tutorial on how to bake a gluten-free cake.

Supervised Fine-Tuning (SFT)



Awesome SFT datasets

Awesome SFT datasets updated 21 days ago

A curated list of interesting datasets to fine-tune language models with.

bjoernp/ultrachat_de
Viewer • Updated Dec 2, 2023 • ↓ 3 • ♡ 4

openchat/openchat_sharegpt4_dataset
Updated Jul 1, 2023 • ↓ 55 • ♡ 138

Note One of the datasets behind OpenChat-3.5

imone/OpenOrca_FLAN
Viewer • Updated Dec 8, 2023 • ↓ 70 • ♡ 10

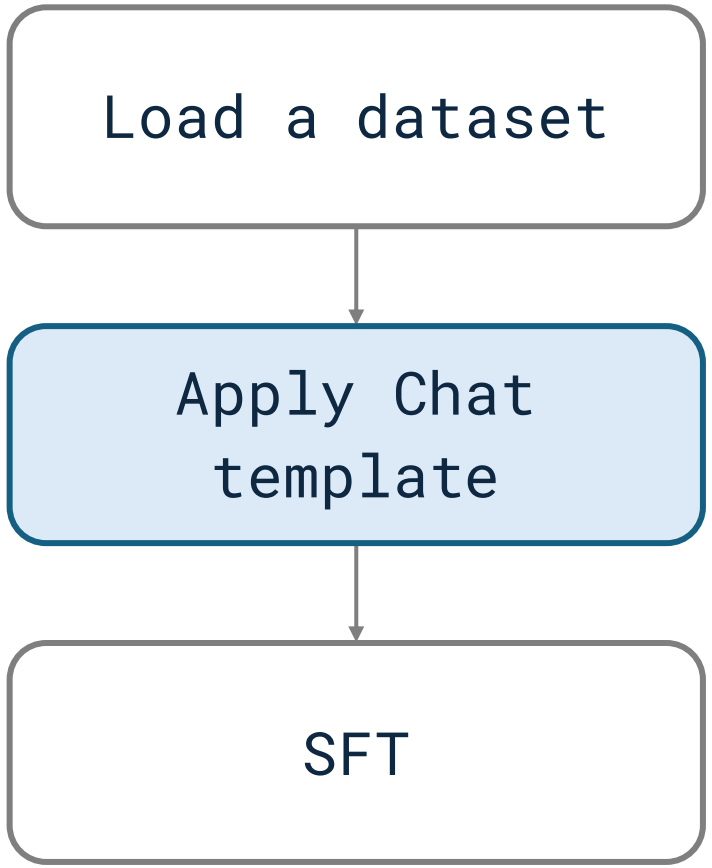
Note One of the datasets behind OpenChat-3.5

tiedong/goat
Viewer • Updated May 26, 2023 • ↓ 154 • ♡ 28

Note One of the datasets behind OpenChat-3.5, focused on arithmetic

LDJnr/LessWrong-Amplify-Instruct
Viewer • Updated Nov 21, 2023 • ↓ 21 • ♡ 36

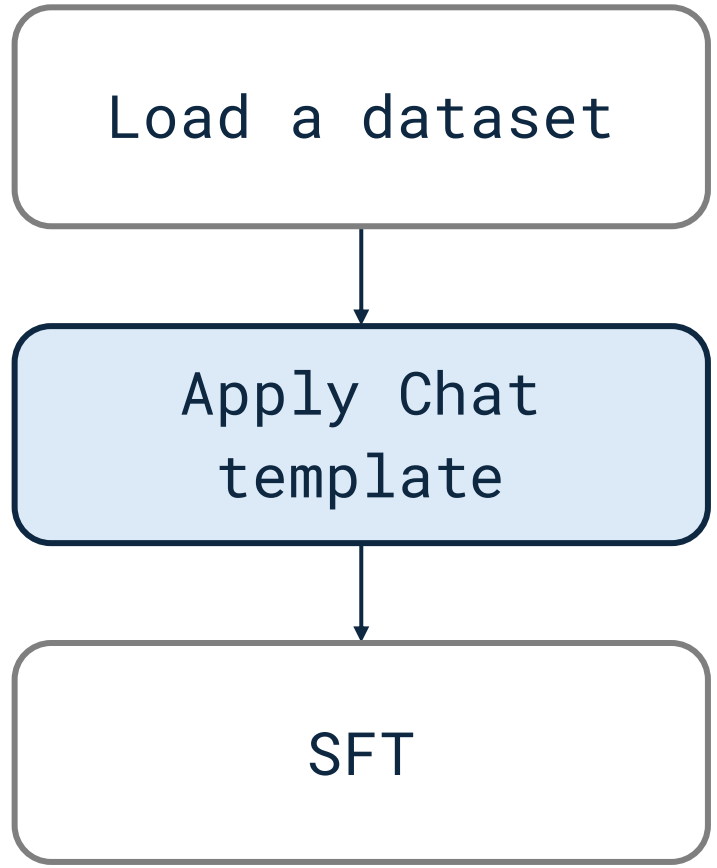
Supervised Fine-Tuning (SFT)



Popular templates:

- ChatML
- Llama-2
- Zephyr

Supervised Fine-Tuning (SFT)



What is 2+2?

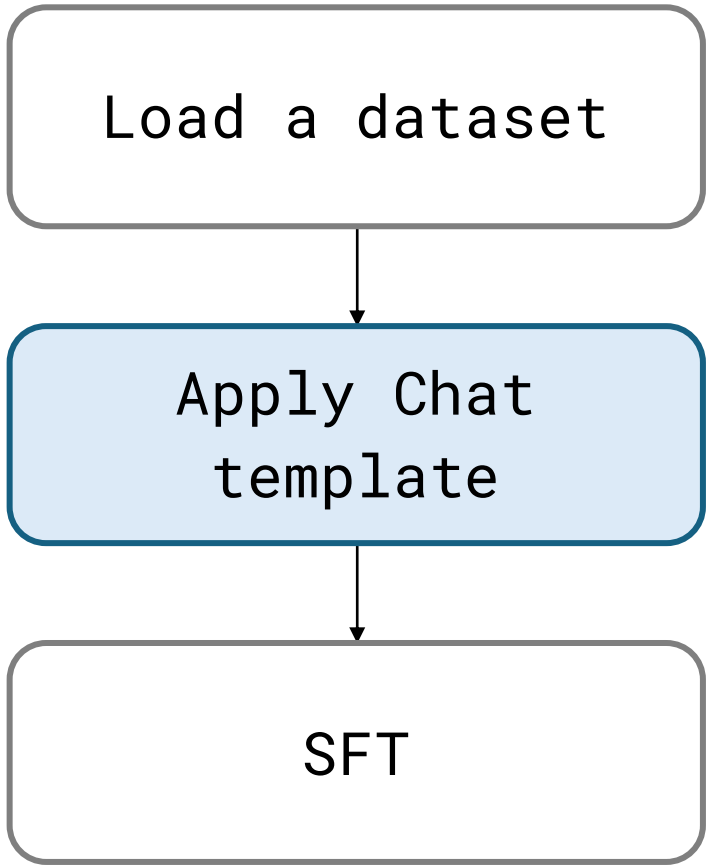
2+2 is equal to 4, how else can I help?

What about 5*7?

5*7 is equal to 35, do you have any further questions?

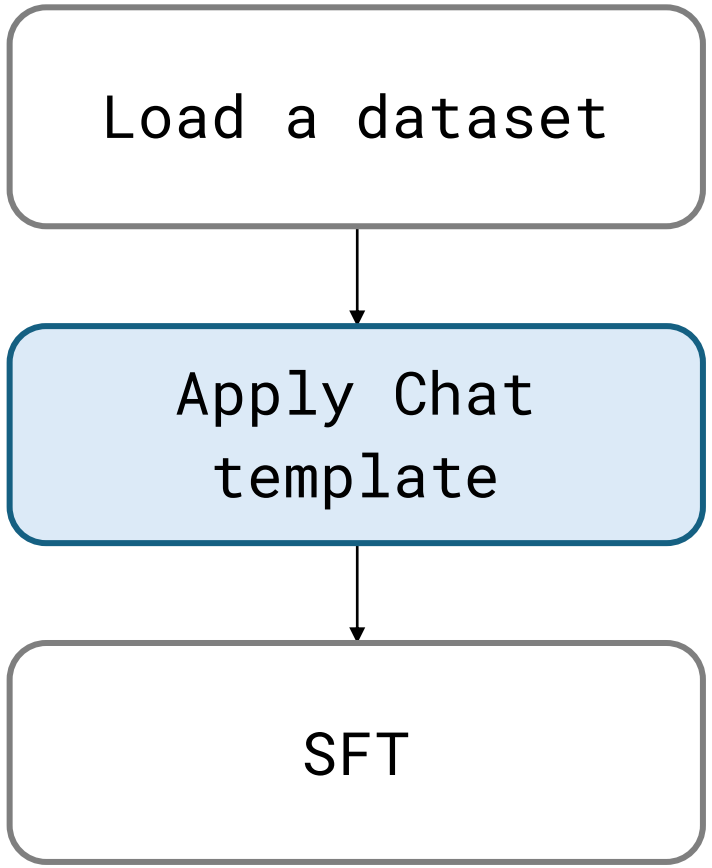
...

Supervised Fine-Tuning (SFT)




```
<|im_start|>system
<|im_end|>
<|im_start|>user
                <|im_end|>
<|im_start|>assistant

     <|im_end|>
<|im_start|>user
                    <|im_end|>
<|im_start|>assistant

                          <|im_end|>
```

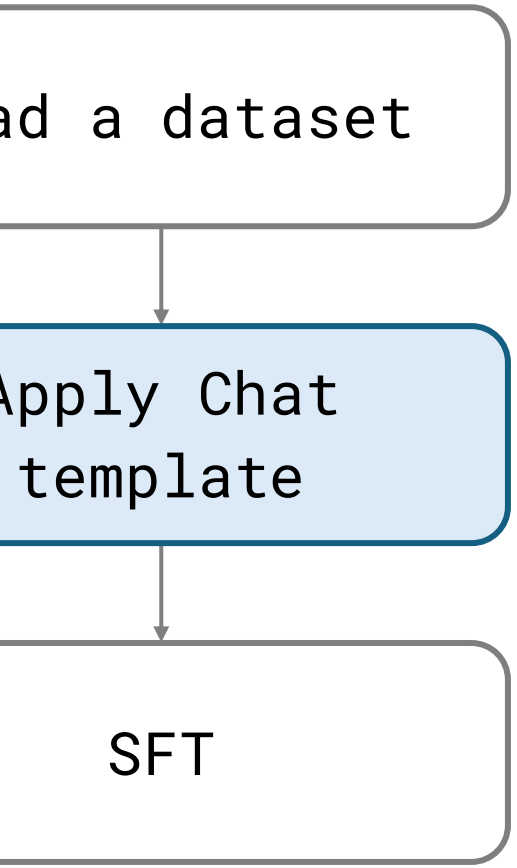
Supervised Fine-Tuning (SFT)



Popular templates:

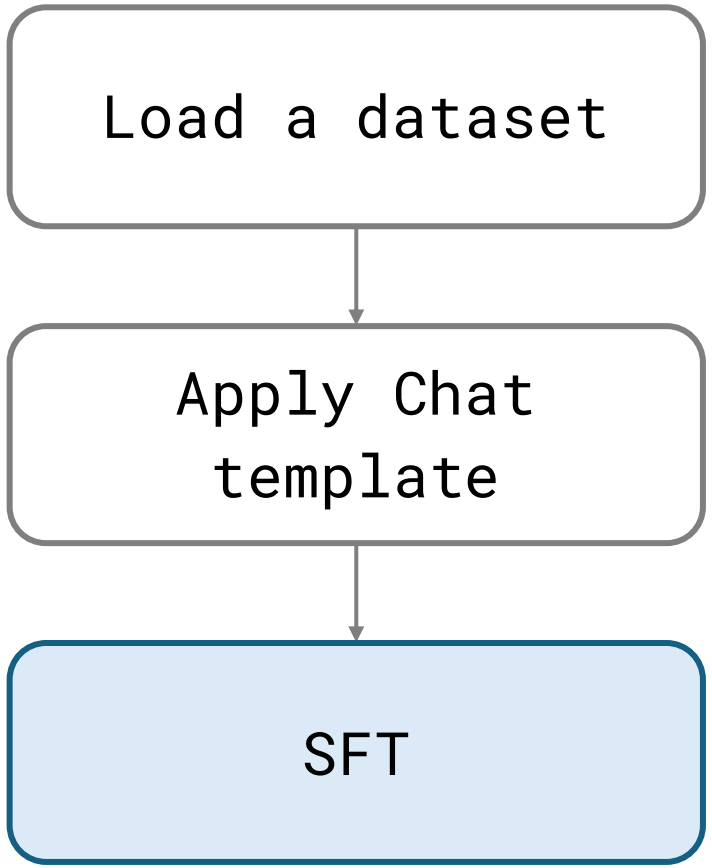
- ChatML <- Recommended
- Llama-2
- Zephyr

Supervised Fine-Tuning (SFT)



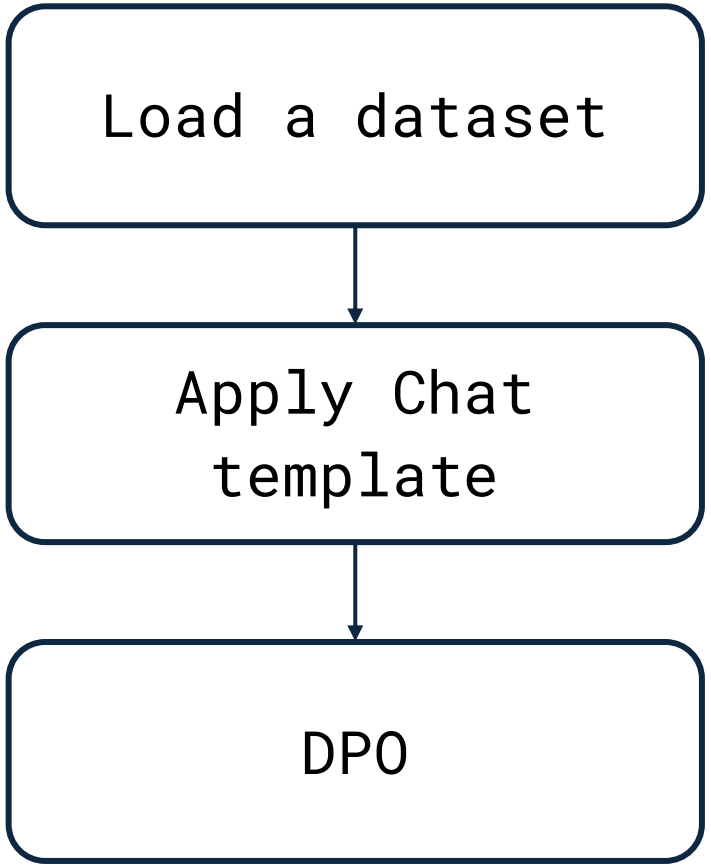

```python
from transformers import AutoTokenizer

model_id = "mistralai/Mistral-7B-v0.1"
tokenizer = AutoTokenizer.from_pretrained(model_id)

CHAT_TEMPLATE = "{% for message in messages %}\n{% if message['role'] == 'user' %}\n{{ '<|user|>\n' +
message['content'] + eos_token }}\n{% elif message['role'] == 'system' %}\n{{ '<|system|>\n' + message['content'] +
eos_token }}\n{% elif message['role'] == 'assistant' %}\n{{ '<|assistant|>\n'  + message['content'] + eos_token
}}\n{% endif %}\n{% if loop.last and add_generation_prompt %}\n{{ '<|assistant|>' }}\n{% endif %}\n{% endfor %}"

tokenizer.chat_template = CHAT_TEMPLATE

def apply_chat_template(example, tokenizer):
    messages = example["messages"]
    # We add an empty system message if there is none
    if messages[0]["role"] != "system":
        messages.insert(0, {"role": "system", "content": ""})
    example["text"] = tokenizer.apply_chat_template(messages, tokenize=False)

    return example

raw_datasets = raw_datasets.map(apply_chat_template, fn_kwargs={"tokenizer": tokenizer})
```

Supervised Fine-Tuning (SFT)

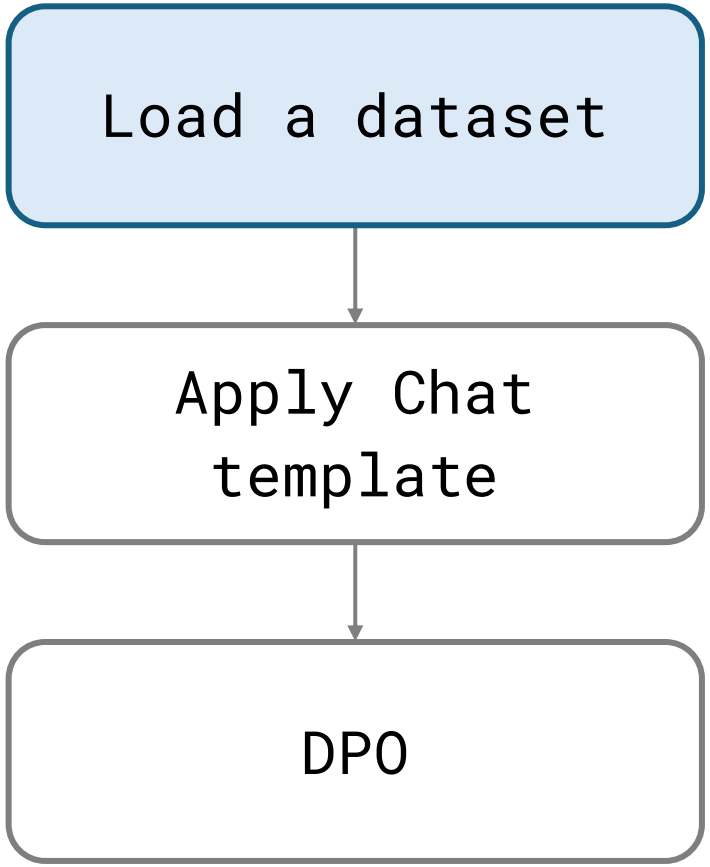



```python
from trl import SFTTrainer

model_id = "mistralai/Mistral-7B-v0.1"

training_args = TrainingArguments(
    learning_rate=2.0e-05,
    gradient_accumulation_steps=4,
    lr_scheduler_type="cosine",
    num_train_epochs=1,
    output_dir=output_dir,
    per_device_eval_batch_size=8,
    per_device_train_batch_size=8,
)
trainer = SFTTrainer(
        model=model_id,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        dataset_text_field="text",
        tokenizer=tokenizer,
        packing=True,
        max_seq_length=tokenizer.model_max_length,
)
trainer.train()
```

Direct Preference Optimization (DPO)



Direct Preference Optimization (DPO)




```
from datasets import load_dataset

raw_datasets = load_dataset("HuggingFaceH4/ultrafeedback_binarized")
print(raw_datasets)
DatasetDict({
    train: Dataset({
        features: ['prompt', 'prompt_id', 'chosen', 'rejected', 'messages', 'score_chosen', 'score_rejected'],
        num_rows: 65120
    })
    test: Dataset({
        features: ['prompt', 'prompt_id', 'chosen', 'rejected', 'messages', 'score_chosen', 'score_rejected'],
        num_rows: 2000
    })
})
```

Direct Preference Optimization (DPO)

Load a dataset

Apply Chat template

DPO

Awesome feedback datasets

Awesome feedback datasets updated Dec 29, 2023

A curated list of datasets with human or AI feedback. Useful for training reward models or applying techniques like DPO.

Anthropic/hh-rlhf
Viewer • Updated May 26, 2023 • ⭳ 55k • ♡ 875

Note The OG of open preference data. Not super great quality

berkeley-nest/Nectar
Viewer • Updated 21 days ago • ⭳ 1.41k • ♡ 172

Note NC

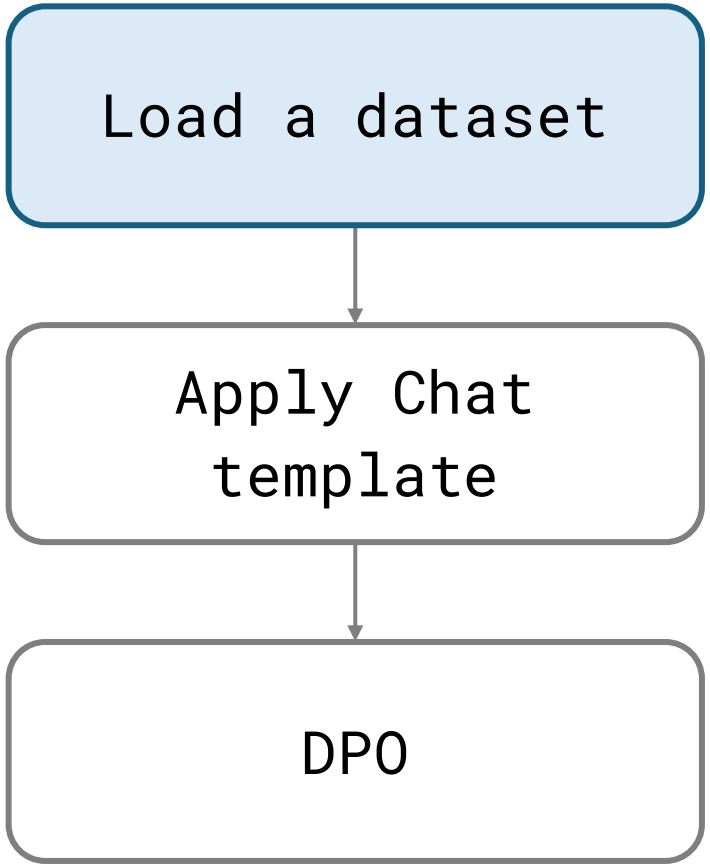
openbmb/UltraFeedback
Viewer • Updated Dec 29, 2023 • ⭳ 5.09k • ♡ 205

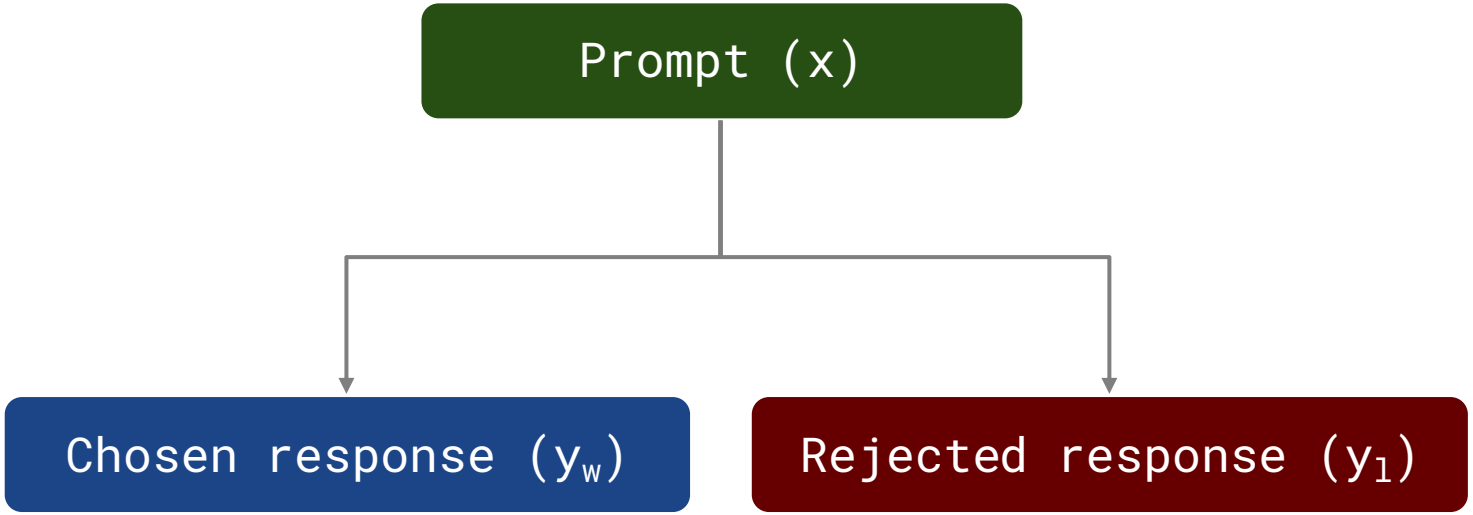
Intel/orca_dpo_pairs
Viewer • Updated Nov 29, 2023 • ⭳ 13.5k • ♡ 160

Note A simple idea to just generate GPT-4 responses and treat them as preferred response.

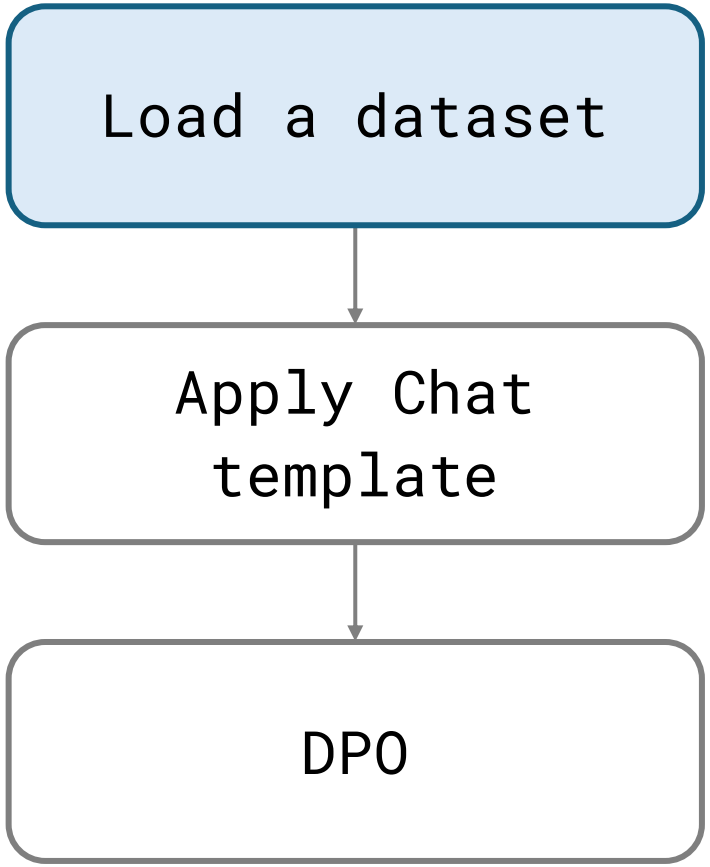
Hello-SimpleAI/HC3
Viewer • Updated Jan 21, 2023 • ⭳ 1.06k • ♡ 154

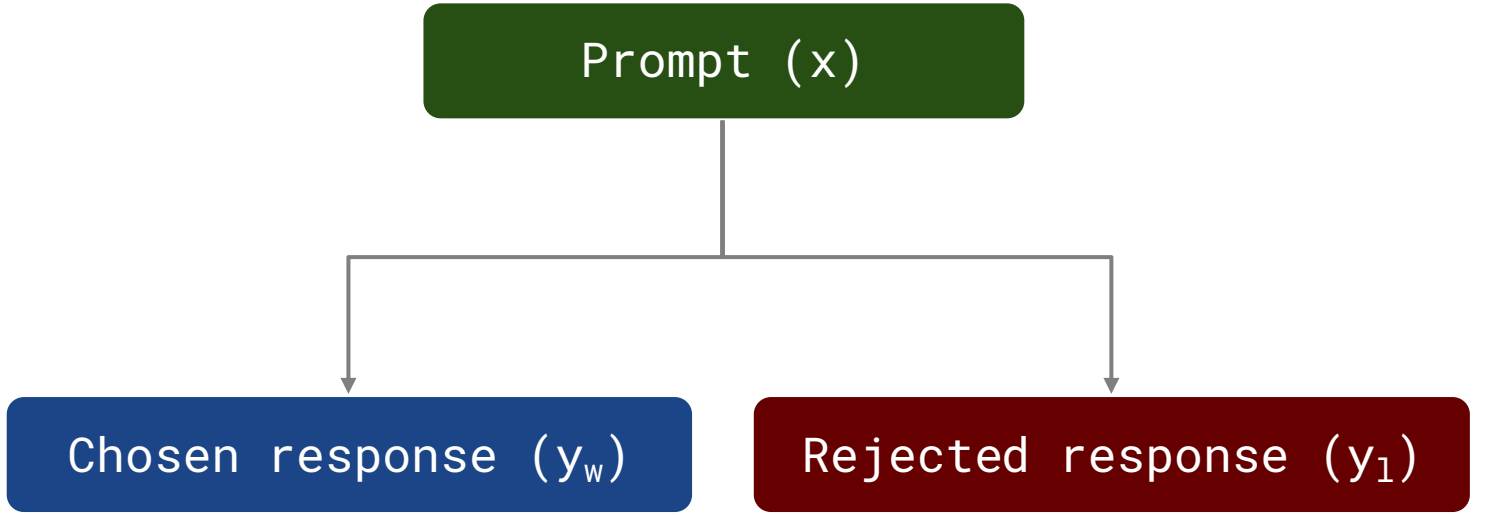
Direct Preference Optimization (DPO)





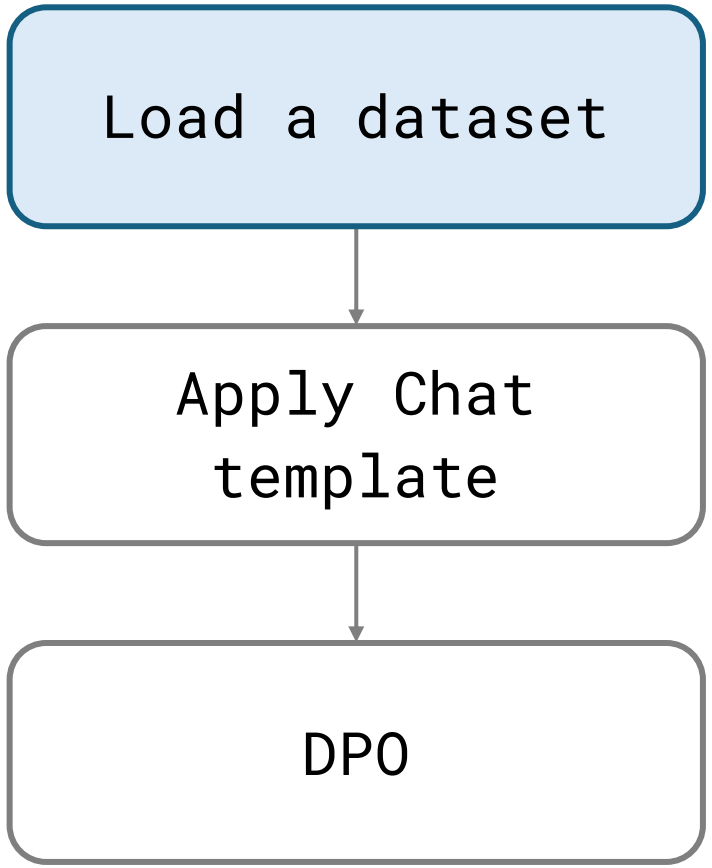
Direct Preference Optimization (DPO)

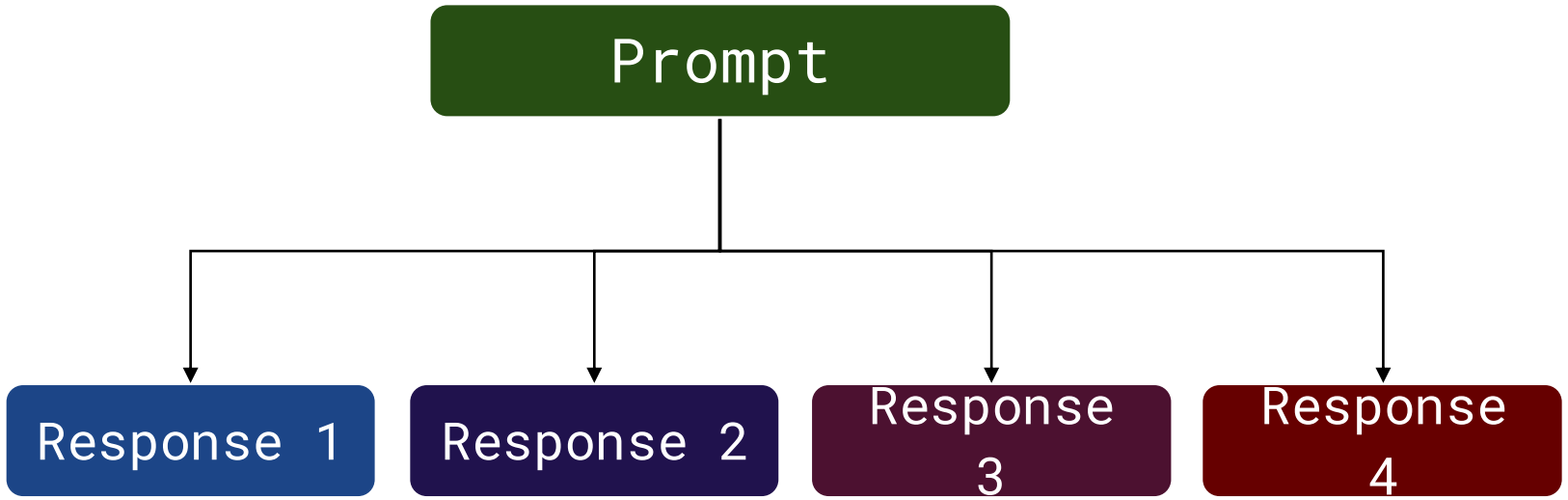




$$\max_{\pi} \mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}} \log\sigma\left(\beta\log\frac{\pi(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Direct Preference Optimization (DPO)

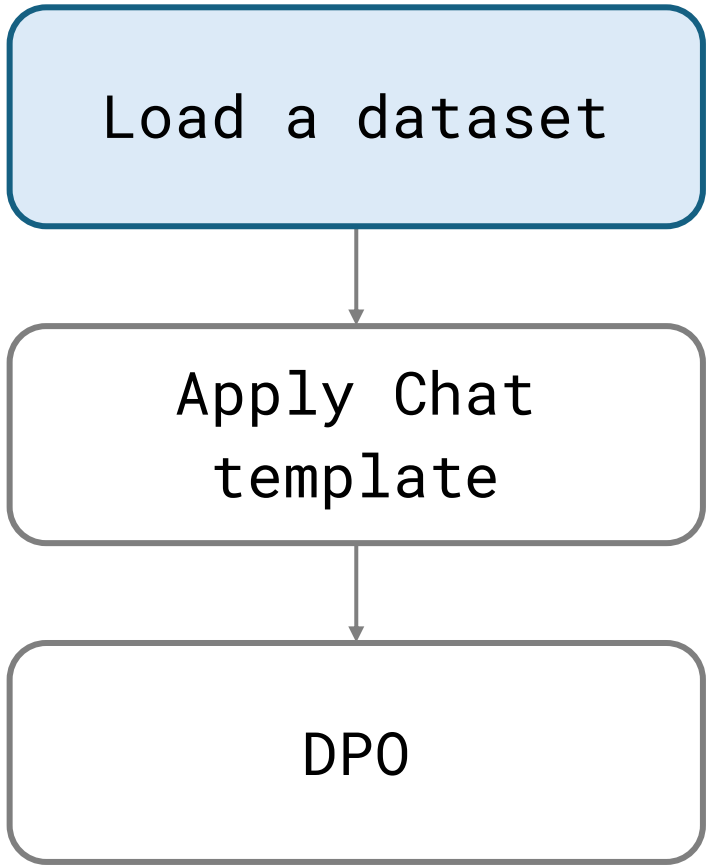


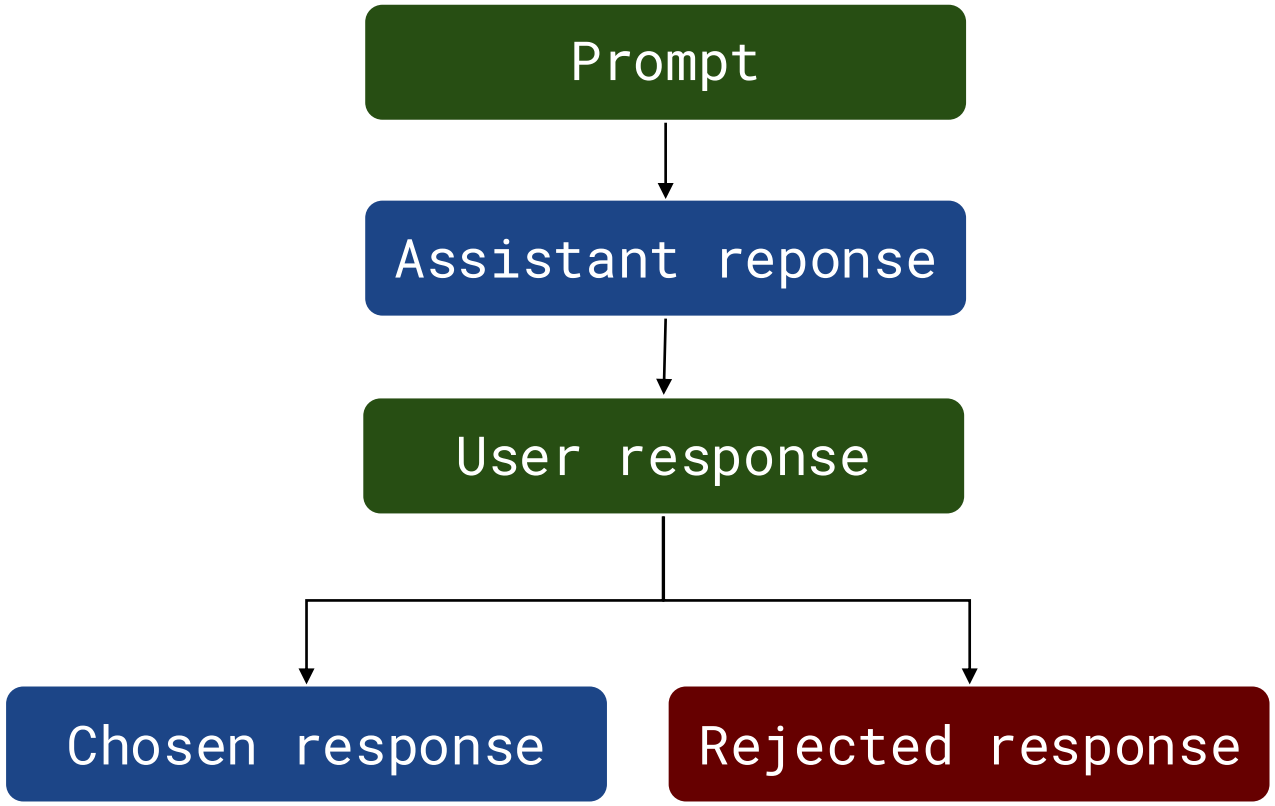


Where:

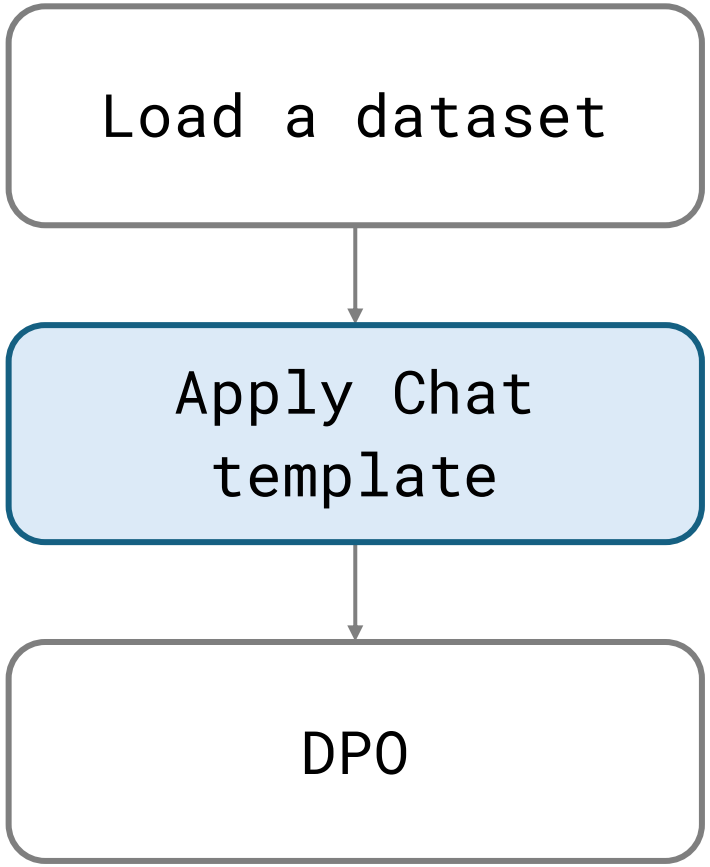
Pref(R1) > Pref(R2) > Pref(R3) > Pref(R4)

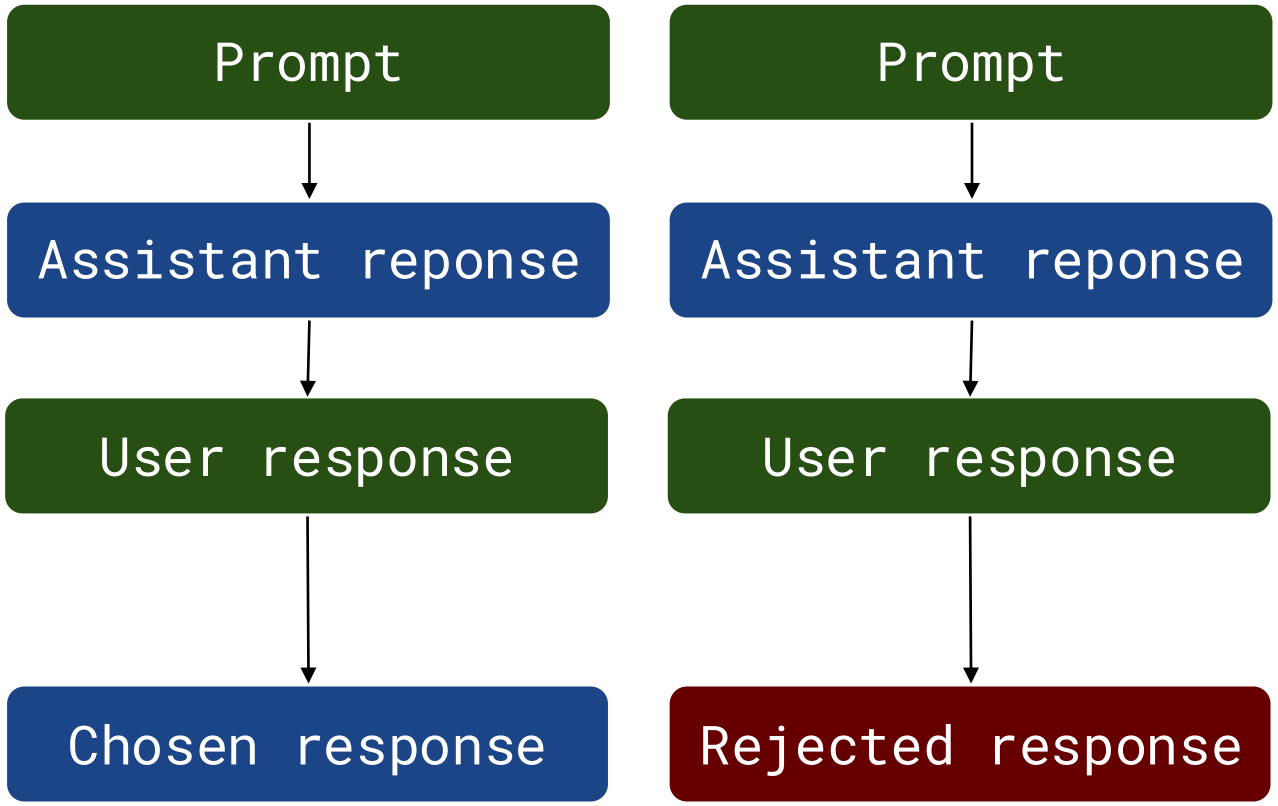
Direct Preference Optimization (DPO)



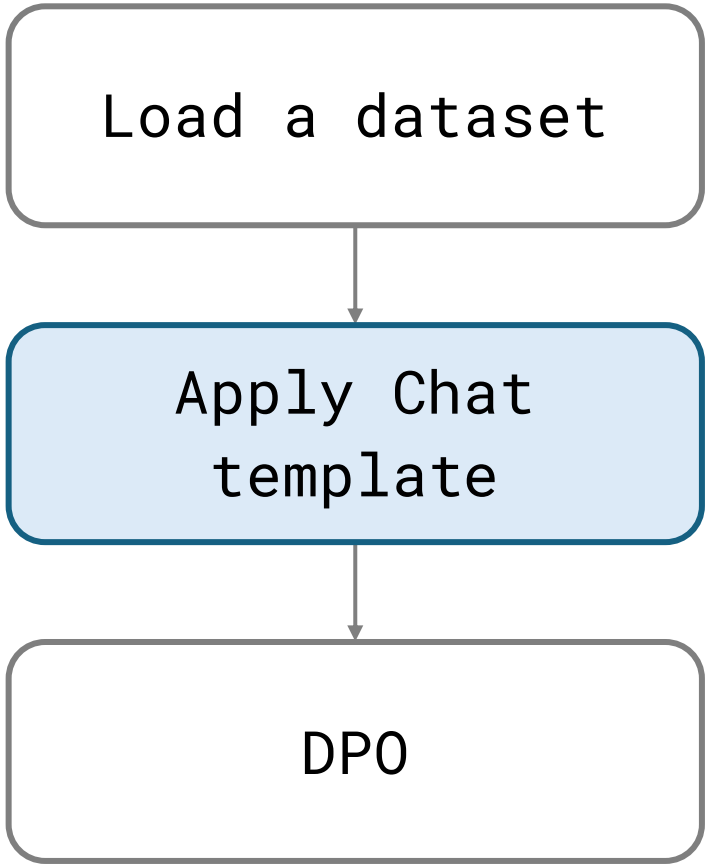


Direct Preference Optimization (DPO)



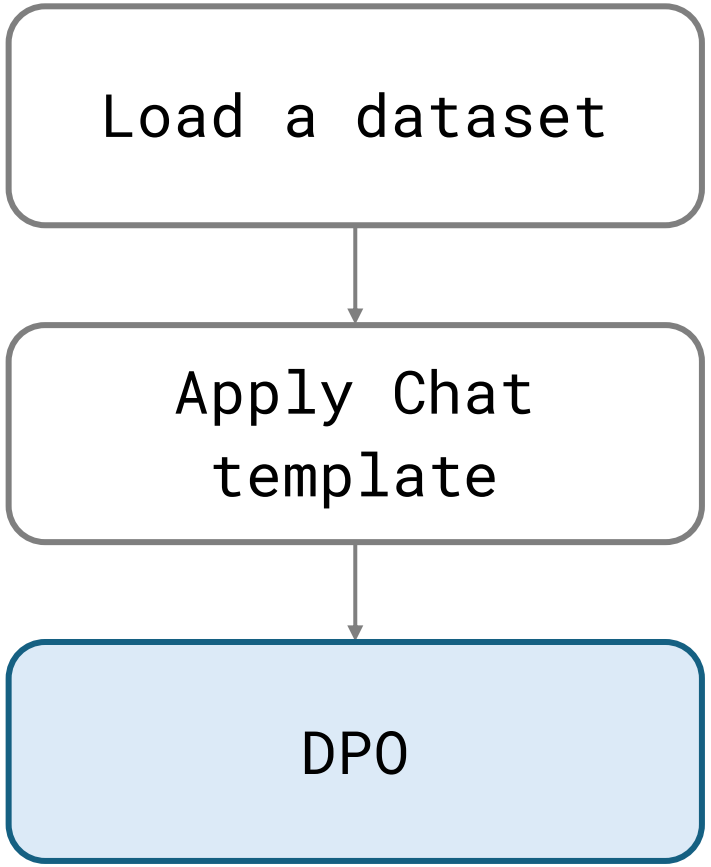


Direct Preference Optimization (DPO)




```python
def apply_chat_template(
    example,
    tokenizer,
):
    prompt_messages = example["chosen"][:-1]
    # Prepend a system message if the first message is not a system message
    if example["chosen"][0]["role"] != "system":
        prompt_messages.insert(0, {"role": "system", "content": ""})
    # Now we extract the final turn to define chosen/rejected responses
    chosen_messages = example["chosen"][-1:]
    rejected_messages = example["rejected"][-1:]
    example["text_chosen"] = tokenizer.apply_chat_template(chosen_messages, tokenize=False)
    example["text_rejected"] = tokenizer.apply_chat_template(rejected_messages, tokenize=False)
    example["text_prompt"] = tokenizer.apply_chat_template(prompt_messages, tokenize=False)

raw_datasets = raw_datasets.map(
        apply_chat_template,
        fn_kwargs={"tokenizer": tokenizer}
)
```

Direct Preference Optimization (DPO)




```python
from trl import DPOTrainer, DPOConfig
from peft import LoraConfig
#from transformers import TrainingArguments

# path where the Trainer will save its checkpoints and logs
output_dir = 'data/zephyr-7b-dpo-lora'

# based on config
training_args = DPOConfig(
    bf16=True,
    beta=0.01,
    do_eval=True,
    eval_steps=100,
    gradient_accumulation_steps=4,
    gradient_checkpointing=True,
    gradient_checkpointing_kwargs={"use_reentrant":False},
    hub_model_id="zephyr-7b-dpo-qlora",
    learning_rate=5.0e-6,
    log_level="info",
    logging_steps=10,
    lr_scheduler_type="cosine",
    max_length=1024,
    max_prompt_length=512,
    num_train_epochs=1,
    optim="paged_adamw_32bit",
    output_dir=output_dir,  # It is handy to append `hub_model_revision` to keep track of your local experiments
    per_device_train_batch_size=4,
    per_device_eval_batch_size=8,
    save_strategy="steps",
    save_steps=100,
    save_total_limit=1,
    seed=42,
    warmup_ratio=0.1,
    loss_type="sigmoid"
)
```

```python
# based on the recipe: https://github.com/huggingface/alignment-handbook/blob/main/recipes/zephyr-7b-beta
peft_config = LoraConfig(
        r=128,
        lora_alpha=128,
        lora_dropout=0.05,
        bias="none",
        task_type="CAUSAL_LM",
        target_modules=["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj",  "up_proj",  "down_proj"],
)
trainer = DPOTrainer(
        model,
        ref_model=None,
        args=training_args,
        train_dataset=raw_datasets["train"],
        eval_dataset=raw_datasets["test"],
        processing_class=tokenizer,
        peft_config=peft_config,
    )
```

Direct Preference Optimization (DPO)

Load a dataset

↓

Apply Chat template

↓

DPO

```python
from trl import DPOTrainer, DPOConfig
from peft import LoraConfig
#from transformers import TrainingArguments

# path where the Trainer will save its checkpoints and logs
output_dir = 'data/zephyr-7b-dpo-lora'

# based on config
training_args = DPOConfig(
    bf16=True,
    beta=0.01,
    do_eval=True,
    eval_steps=100,
    gradient_accumulation_steps=4,
    gradient_checkpointing=True,
    gradient_checkpointing_kwargs={"use_reentrant":False},
    hub_model_id="zephyr-7b-dpo-qlora",
    learning_rate=5.0e-6,
    log_level="info",
    logging_steps=10,
    lr_scheduler_type="cosine",
    max_length=1024,
    max_prompt_length=512,
    num_train_epochs=1,
    optim="paged_adamw_32bit",
    output_dir=output_dir,  # It is ha
    per_device_train_batch_size=4,
    per_device_eval_batch_size=8,
    save_strategy="steps",
    save_steps=100,
    save_total_limit=1,
    seed=42,
    warmup_ratio=0.1,
    loss_type="sigmoid"
)
```

$$\left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right)\right]$$

More on beta and Alignment losses: https://huggingface.co/blog/pref-tuning

DPO Training tips

Beta: test from 0.01 - 1.0

Learning rate: much smaller than for SFT ~100x smaller (5E-7)

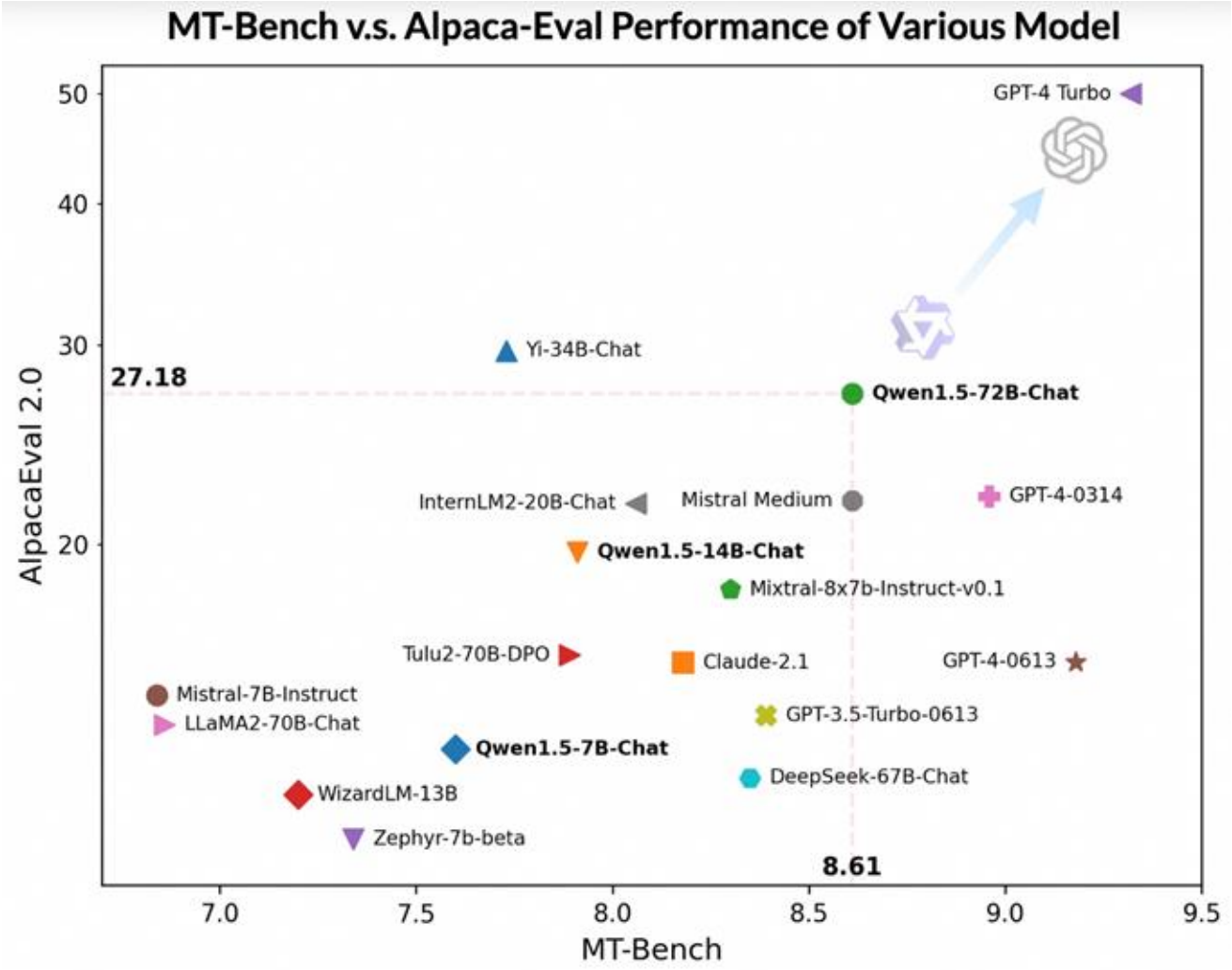
Batch size: tradeoff between global batch size and n epochs

Optimizer: Adam appears better than RMSProp

Scheduler: Cosine > Linear

The best SFT model != Best DPO model

LoRA: Appears to regularize the model compared to full fine-tune



https://qwenlm.github.io/blog/qwen1.5

Evaluating Chatbots

- OpenLLM LeaderBoard - Not Chatbot focused, leakage, overfitting
- MT Bench - Usage
- Alpaca Eval - Usage
- LLamaindex (RAG)
- Human Eval - Lmsys Chatbot Arena

Definición del tamaño de los LLMs

- El tamaño de un modelo LLM se refiere al **número de parámetros** que contiene
- Los parámetros son como "perillas e interruptores" que el modelo ajusta durante el entrenamiento para aprender patrones en los datos
- **Más parámetros generalmente conducen a una mejor comprensión contextual y salidas más complejas**

Categorías de Tamaño de Modelos LLM

Modelos Pequeños: Menos de 1 mil millones de parámetros.	• Ejemplos: **BERT Base** (110M), **GPT-2 Small** (117M).
Modelos de Tamaño Mediano: 1–10 mil millones de parámetros.	• Ejemplos: T5, GPT-3 Ada.
Modelos Grandes: 10+ mil millones de parámetros.	• Ejemplos: **GPT-3 (175B)**, **GPT-4**.

¿Siempre son Mejores los Modelos Más Grandes?

- Si bien aumentar el tamaño generalmente mejora el rendimiento, existen **rendimientos decrecientes.**
  - **GPT-2 Small (117M):** Comprensión contextual limitada, generación de texto simple.
  - **BERT Base (110M):** Buenos embeddings de texto, clasificación, análisis de sentimiento.
  - **Mistral 7B (7.3B):** Alto rendimiento con bajos recursos computacionales.
  - **GPT-3 (175B):** Razonamiento fuerte, tareas versátiles.
  - **GPT-4 (>1T estimado):** Razonamiento a nivel humano, multimodal, resolución de problemas complejos.
  - **Claude 3.5 Sonnet:** Alto rendimiento en tareas de codificación, multilingües y de razonamiento.

Tamaño del Modelo	Mejora del Rendimiento	Compromisos
Pequeño a Mediano	Significativa	Uso de recursos moderado
Mediano a Grande	Marginal	Alto impacto en cómputo y costo
Muy Grande (>100B)	Mínima	Requisitos de recursos extremos

Ventajas y Desventajas: Modelos Pequeños vs. Grandes

Modelos Pequeños:
Ventajas:
Inferencia más rápida y menor latencia.
Más rentables y energéticamente eficientes.
Más fáciles de escalar.
Implementables en dispositivos con recursos limitados.
Adecuados para tareas simples como clasificación de texto, análisis de sentimiento en textos cortos.
Buenos para reconocimiento automático del habla.
Desventajas:
Comprensión contextual limitada.
Pueden "olvidar" puntos clave en textos largos.
Razonamiento generalmente más débil.
Menos versátiles para tareas complejas.

Modelos Grandes:
Ventajas:
Mejor comprensión de matices en el lenguaje (idiomas, metáforas).
Manejan ventanas de contexto más grandes.
Mejor razonamiento y generalización.
Mejores en tareas de "zero-shot".
Excelentes para tareas de propósito general como generación de IA, traducción.
Desventajas:
Mayor costo computacional.
Mayor impacto ambiental.
Requieren infraestructura especializada (GPUs/TPUs).
Inferencia más lenta y mayor latencia

El Punto Medio: Finetuned models

- Combinan la eficiencia de modelos más pequeños con mejoras de rendimiento específicas para una tarea
- Se parte de un modelo pre-entrenado (a menudo grande) y se ajusta en un conjunto de datos más pequeño y específico de un dominio
- **Ventajas:** Equilibrio entre rendimiento y eficiencia
- **Casos de Uso:** Análisis de texto específico de un dominio (legal, médico, financiero), optimización de respuestas de atención al cliente, análisis de sentimiento e intención personalizados

Eligiendo el Modelo LLM Adecuado

- **Tarea:** ¿Es simple o compleja?
- **Recursos Disponibles:** Presupuesto, hardware, memoria
- **Requisitos de Rendimiento:** Velocidad, precisión, latencia
- Complejidad promedio de las consultas
- Requisitos de latencia

Estrategia recomendada

- Comenzar con un modelo más pequeño
- Probar su rendimiento
- Escalar según sea necesario
- Considerar el ajuste fino para optimizar el rendimiento en tareas específicas

Eligiendo el Modelo LLM Adecuado

Tamaño del Modelo	Ventajas	Casos de Uso Comunes
Pequeños	Rápido, rentable, poca memoria	Clasificación de texto, análisis de sentimiento (textos cortos), chatbots básicos, aplicaciones integradas, reconocimiento automático del habla.
Medianos	Buen equilibrio	(Implícito: muchas tareas generales con eficiencia razonable)
Grandes	Razonamiento avanzado, versátil	Generación de IA (textos largos, código), traducción compleja, razonamiento contextual.
Ajustados	Optimización específica	Análisis de texto de dominio específico, soporte al cliente automatizado, análisis de sentimiento personalizado.