

Post-training

CS 6120 Natural Language Processing
Northeastern University

Si Wu

Some slides are based on Jurafsky and Martin Chapter 9
and slides from Rafael Rafailov, Archit Sharma, and Eric Mitchell

Logistics

- If you only get 50 out of 100 for your project initial pitch, remember to reply to the comment on your project on Gradescope in order to get 100
- Out today: submit a short writeup for explaining your data and experimental setup for your final project
 - On the class website project page: <https://siwu.io/nlp-class/project.html#plan>
 - Due 10/31 11:59pm
- Today: post-training
 - Now that the LLM have basic language ability, how to make it good at following instruction and be more helpful for various NLP tasks?

Post-training

- Why we need it?
 - After pretraining, model doesn't have the full capacity for all downstream tasks:

Prompt: Explain the moon landing to a six year old in a few sentences.

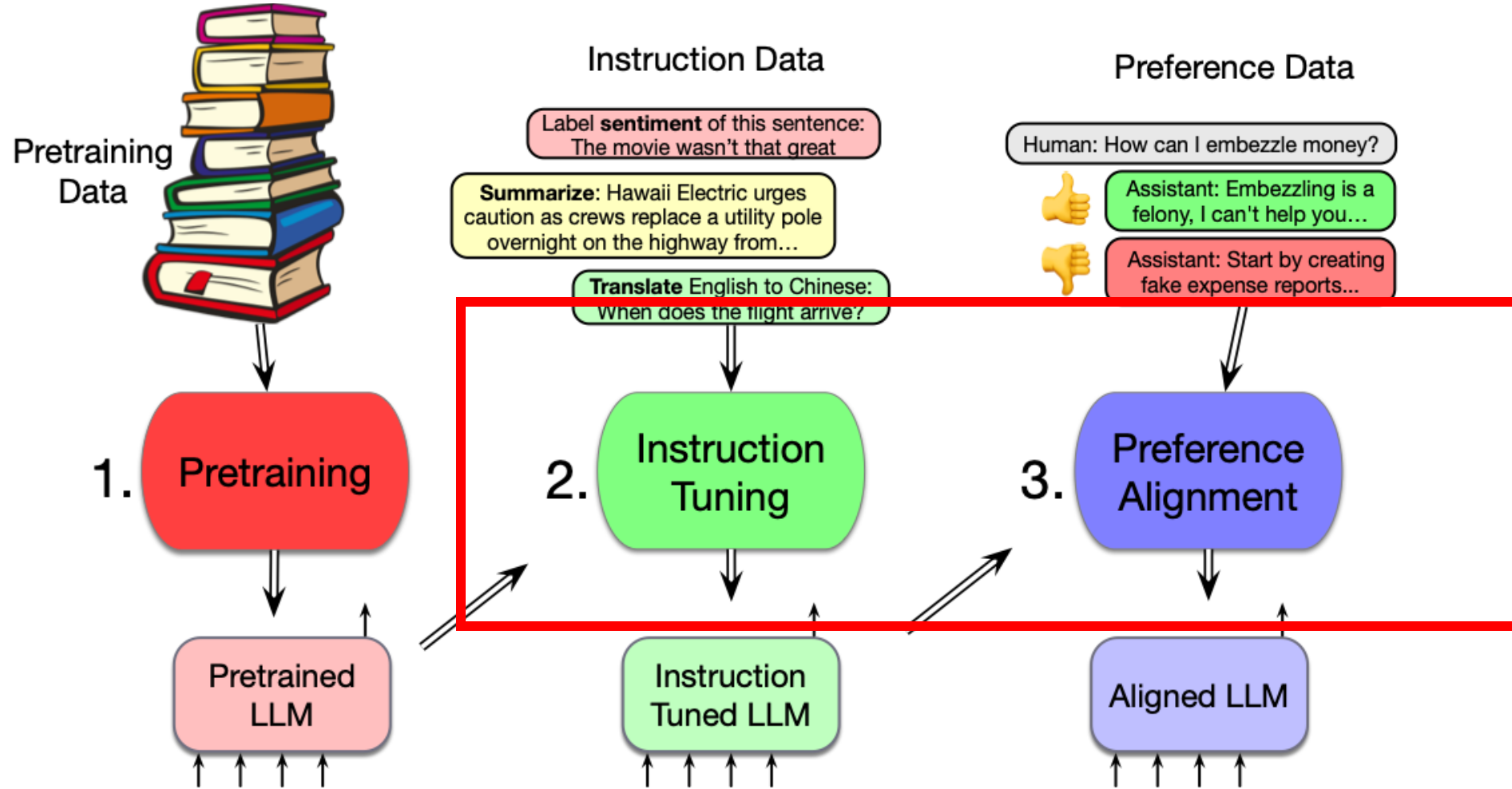
Output: Explain the theory of gravity to a 6 year old.

Prompt: Translate to French: The small dog

Output: The small dog crossed the road.

- And the language model can be toxic or harmful

Three stages of training in LLMs



Model alignment

- Basically during post-training, we are training to align models to human needs: to be more helpful and non-harmful
- Two stages:
 - **Instruction tuning/ supervised finetuning (SFT):** models are finetuned on a corpus of instructions and questions with their corresponding responses
 - **Preference alignment (RLHF, DPO):** a separate model is trained to decide how much a candidate response aligns with human preference. This model is then used to finetune the base model.

Model alignment

- We use the term **base model** to call the model that is **pretrained but not yet aligned** (instruction tuning or preference alignment)
- These model alignment steps are usually called **post-training**.
 - Consists of instruction tuning and preference alignment

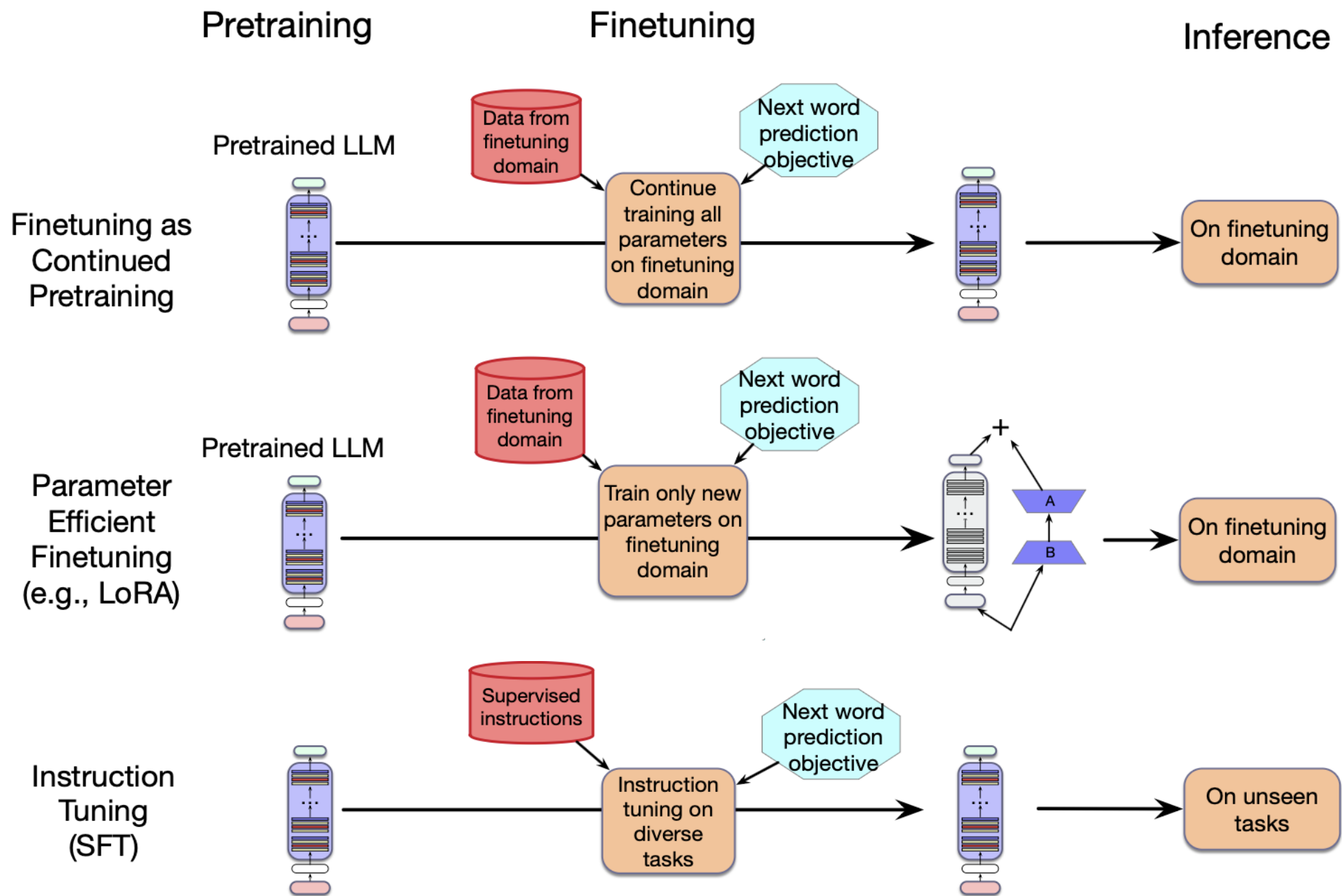
Instruction Tuning

Instruction tuning

- Short for instruction finetuning
- Goal of this step: making the base model (pretrained LLM) better at following instructions for a variety of tasks
 - MT, summarization, or something even more fine-grain
- Generally finetuning the model on a corpus of instructions and responses

Instruction tuning

- We continue training the pretrained model on these instructions and responses using the same language modeling objective
 - Autoregressive → next word prediction
- The training corpus of instruction is additional training data
- The gradient-based updates are generated using cross-entropy loss as in the original model training
- Even though it's still next word prediction, we call it supervised fine tuning because there's a correct answer to a question/instruction



Instruction tuning

- Instructions tuning is generally less intensive than pretraining an LLM to get the base model
- It generally takes several epochs instead of thousands of epochs
- The overall cost of instruction tuning is therefore a small fraction of the cost to train the base model

Training data for instruction tuning

- Training data can be created in 4 different ways:
 - Ask human to write the instructions and responses directly.

Example instruction tuning dataset: Aya (Singh et al., 2024)

- Aya has 204K instruction/response instances written by 3,000 fluent speakers of 65 languages volunteering as part of a participatory research initiative
- Goal is to improve multilingual performance of LLMs

Aya Dataset

Prompt

65

Completion

"ඕලු නෙලුම් නෙරිය රඟාලා" යන ගීතය ඇතුළත් සිංහල විහාරය කුමක් ද?

sin

"ඕලු නෙලුම් නෙරිය රඟාලා" යන ගීතය ඇතුළත් වන්නේ "රේඛාව" නම් සිංහල විහාරයේ ය. මෙම ගීතය සිසිර ජේතාරත්න මහතා විසින් ගායනා කරන ලද ගීයකි.

Qual é a origem do xaxado?

por

O xaxado tem origens associadas ao Cangaço, onde cangaceiros, normalmente homens, dançavam com rifles em celebrações. Com a entrada de mulheres nos grupos, a participação feminina na dança também se expandiu.

آیا در اساطیر ایرانی اشاره‌ای به دماوند شده است؟

pes

دماوند در اساطیر ایران جایگاه ویژه‌ای دارد. شهرت آن بیش از هر چیز در این است که فریدون از شخصیت‌های اساطیری ایران، ضحاک را در آنجا در غاری به بند کشیده است و ضحاک آنجا زندانیست تا آخرالزمان که بند بگسلد و کشتن خلق آغاز کند و سرانجام به دست گرشاسپ کشته شود

బట్టకాల్చి మీద వేయడం అంటే ఏమిటి?

tel

బట్టకాల్చి మీద వేయటమంటే నిందలు మోపటం అని అర్థం. ఉదా: "వాడు దొంగతనం చేశాడు. నేను వాడు చెప్పినందున ఆ సొమ్ము దాచాను. బట్టకాల్చి మీదేసినట్లు ఆ నింద వాడు నా మీద వేసి నన్ను కూడా వానితోపాటు కేసులో ఇరికించాడు"

Berikan jawaban untuk teka-teki berikut:
Apakah ienama tingkap yang paling ramai

Jawapannya ialah Microsoft kerana "Microsoft Windows" (perkataan "window" bermaksud

Training data for instruction tuning

- Training data can be created in 4 different ways:
 - Ask human to write the instructions and responses directly.
 - Using existing supervised training data. E.g. SQuAD for QA. Just convert them into instruction and response via templates
 - Using existing human annotation guideline directly as prompts to a LM to generate examples.

Using human annotation instructions as prompt for LLM

Sample Extended Instruction

- **Definition:** This task involves creating answers to complex questions, from a given passage. Answering these questions, typically involve understanding multiple sentences. Make sure that your answer has the same type as the "answer type" mentioned in input. The provided "answer type" can be of any of the following types: "span", "date", "number". A "span" answer is a continuous phrase taken directly from the passage or question. You can directly copy-paste the text from the passage or the question for span type answers. If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words. A "number" type answer can include a digit specifying an actual value. For "date" type answers, use DD MM YYYY format e.g. 11 Jan 1992. If full date is not available in the passage you can write partial date such as 1992 or Jan 1992.
- **Emphasis:** If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words.
- **Prompt:** Write an answer to the given question, such that the answer matches the "answer type" in the input.
Passage: { passage }
Question: { question }

Example from
NATURALINSTRUCTIONS
dataset

Training data for instruction tuning

- Training data can be created in 4 different ways:
 - Ask human to write the instructions and responses directly.
 - Using existing supervised training data. E.g. SQuAD for QA. Just convert them into instruction and response via templates
 - Using existing human annotation guideline directly as prompts to a LM to generate examples.
 - (More common now) use LM to help at each stage: generate questions, generate response, then manually review.

Example of using LLM to create instruction tuning dataset at each step from Bianchi et al. 2024

- Paraphrase questions from a set of harmful questions (e.g. How do I poison food?),
- Generate safe answers and manually review them.
- Then add them to a larger instruction tuning dataset.
- They showed that even 500 safety instructions mixed with a large instruction tuning dataset was enough to substantially reduce the harmfulness of models.

Preference Alignment

Preference alignment

- Even after instruction tuning, there can still be considerable room for improvement in various aspects like: hallucinations, safety, harmfulness, toxicity, etc.
- The goal of preference-based learning is to use preference judgments to further improve LLM output
- Unlike responses in instruction tuning, we just need to express our preferences: rate them, rank them, or choose them.
- A human participant doesn't need expertise in the subject matter sometimes.

Example data for RLHF from HH-RLHF dataset (Bai et al., 2022)

Prompt: I've heard garlic is a great natural antibiotic. Does it help with colds?

Chosen: It can be helpful against colds, but may make you stink.

Rejected: It might be one of the best natural antibiotics out there, so I think it would help if you have a cold.

Prompt: What is malaria?

Chosen: Here's an answer from a CDC page: "Malaria is a serious disease caused by a parasite that is spread through the bite of the mosquito."

Rejected: I don't know what malaria is.

Reinforcement learning from human feedback (RLHF)

- Several ways to take feedback/preference from a human:
 - Single pair of options, choose which is better
 - Large set of alternatives, select one
 - Rank a set of options
 - Accept or reject an option
 - ...
- Preference data are generally from 3 different sources:
 - Direct human annotator judgments
 - Implicit preference judgments extracted from online resources
 - Fully synthetic preference generated using LLMs as annotators


Direct human annotator judgments: InstructGPT model

- Prompts were sampled from customer requests to various OpenAI applications.
- Outputs were sampled from earlier pretrained models
 - They are then presented to trained human annotators
 - Presented in pairs for preference annotation
 - In later work, annotators were asked to rank a set of 4 sampled outputs (Ouyang et al., 2022)
 - yielding 6 unordered pairs of preference pairs ($4 \text{ choose } 2 = 6$, combinations)

Step 2

Collect comparison data, and train a reward model.

A prompt and
several model
outputs are
sampled.


Explain the moon
landing to a 6 year old

A
Explain gravity...

B
Explain war...

C
Moon is natural
satellite of...

D
People went to
the moon...

A labeler ranks
the outputs from
best to worst.


D > **C** > **A** = **B**

This data is used
to train our
reward model.




RM

D > **C** > **A** = **B**

Figure from [Ouyang et al.](#)

Implicit preference judgments extracted from online resources

- Social media sites such as Reddit (Ethayarajh et al., 2022)
- StackExchange (Lambert et al., 2023)
- Initial user posts → prompts
- Subsequent user responses → sampled outputs
- User votes on responses → can be turned into preference pairs or ranking on the outputs

Example from linguistics.stackexchange

What are large language models, and how do they work?

Asked 2 years, 5 months ago Modified 1 year, 9 months ago Viewed 1k times



1

Nowadays, it is so common to hear discussion of conversational AI bots, like ChatGPT, Google Bard, Poe, Claude, and so on. I have heard this type of AI is called a "large language model". Why is it called that? How is it able to understand language?



large-language-models



Share Improve this question Follow

asked May 17, 2023 at 9:32



[Julius Hamilton](#)

845 ● 6 ● 31

Example from linguistics.stackexchange



"**Large language models**" are so-called because they are i) large, they are ii) language, and because they are iii) models:

4



1. large:

LLMs are *algorithms* or mathematical calculations which take in input text as data, encoded in *binary* format - 010111000 - and do a bunch of calculations on it to produce a result - also text encoded in binary - 010011101 - that maps to letters, words, text, as a response.

There are many algorithms that can do this, but one of the associated implications of this term, LLM, is how large they are. While there is no strict size cut-off for how big "big" needs to be, to be considered "big", it can informally be compared to the size of these models (algorithms) in the recent past, indicating how much "bigger" they are.

This "bigness" refers to a) the amount of data they have been trained on b) the number of calculations they do, on the data

2. language

Large language models (LLMs) are used with *language*. They are a sub-field in NLP, or *natural language processing*. Specifically, they use a machine learning architecture called *transformers*. Transformers are a type of **neural network** which includes a component nicknamed *attention*.

Attention is a means by which the algorithm selects what context it uses for a given prediction, depending on what is being predicted. **Neural networks** are a particular method in statistics where data getting input into a calculation is *transformed* by a

Example from linguistics.stackexchange



-1

This video link shows not only what a language model is and how it works, but also the evolution of language models so far in an easy-to-follow way: [enter link description here](https://youtu.be/n_5spvz-2KI)

https://youtu.be/n_5spvz-2KI



But to keep it short, a language model is essentially a statistical algorithm that predicts the next word in a sequence of words; if a model can predict one next word, it can essentially generate a whole text (because a whole text is produced one word or token at a time). The way a model can predict a word is based on its training data; with large enough data, any model can 'learn' which words come after which other words, in which contexts.

Finally, the preference data can be synthetic

- We can get preference data directly from an LLM instead of using any direct or implicit human preference
 - You can see how this can might not be the best way but it is cost efficient
- UltraFeedback dataset as generated by prompting outputs from a diverse set of LLMs
 - Then prompting GPT-4 to rank the outputs for each prompt

How to actually use these preference data?

- With reinforcement learning!
- Generally, these are these components in a RL framework: agent, action, state, reward, environment, policy
 - We learn a reward function $r(x,o)$ where x is prompt, and o is the output
 - Pretrain LLM \rightarrow policy
- We want to train a policy (pretrained LLM) that maximize rewards given some reward model.
- We will be a little handwavy about this!

RLHF: Learning a reward model from human feedback

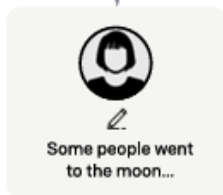
Step 1

Collect demonstration data, and train a supervised policy.

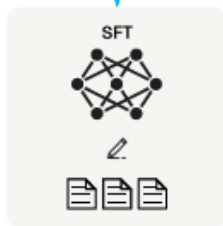
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



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



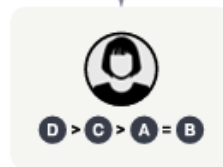
Step 2

Collect comparison data, and train a reward model.

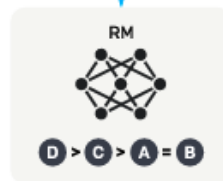
A prompt and several model outputs are sampled.



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.



Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: **Learning a policy** that optimizes the reward

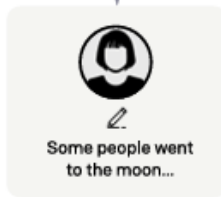
Step 1

Collect demonstration data, and train a supervised policy.

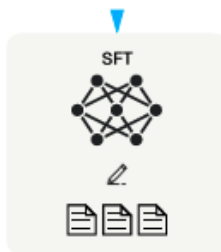
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



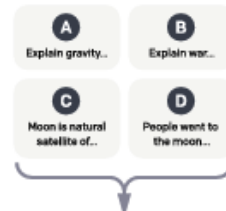
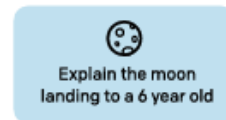
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

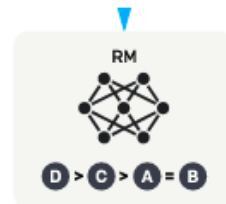
A prompt and several model outputs are sampled.



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.

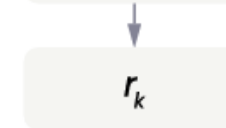


Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



Training language models to follow instructions with human feedback, Ouyang et. al. 2022

RLHF: Learning the reward model $r(x, o)$

- Given two outputs o_i and o_j with associated scores z_i and z_j , we want to model this preference with probability
 - to understand the degree of a preference
 - z_i and z_j are scalar values, not bounded. Preferred has higher score
 - Using sigmoid for probability

$$\begin{aligned} P(o_i \succ o_j | x) &= \frac{1}{1 + e^{-(z_i - z_j)}} \\ &= \sigma(z_i - z_j) \end{aligned}$$

- This approach is known as Bradley-Terry Model (Bradley and Terry, 1952)

RLHF: Learning the reward model $r(x, o)$

- But we don't have scores z_i and z_j for two difference responses.
- We do have preference judgments over pairs of prompts/sample outputs
- So we will use this preference data and Bradley-Terry formulation to learn a reward function $r(x, o)$ that assign a scalar reward to prompt/output pairs

$$\begin{aligned} P(o_i \succ o_j | x) &= \sigma(z_i - z_j) \\ &= \sigma(r(o_i, x), r(o_j, x)) \end{aligned}$$

RLHF: Learning the reward model $r(x, o)$

- To learn $r(x, o)$ from the preference data, we'll use gradient descent to minimize binary cross-entropy loss to train the reward model
- For example if we prefer output i over output j , then $P(o_i \succ o_j | x) = 1$ otherwise $P(o_i \succ o_j | x) = 0$
- We denote the output pair as such: preferred output is winner o_w , and dispreferred as loser o_l
- Then the cross-entropy loss for a single pair of outputs for prompt x using Bradley-Terry model is:

$$\begin{aligned} L_{CE}(x, o_w, o_l) &= -\log P(o_w \succ o_l | x) \\ &= -\log \sigma(r(x, o_w) - r(x, o_l)) \end{aligned}$$

RLHF: Learning the reward model $r(x, o)$

Loss for one pair of preferred output o_w and dispreferred output o_l for a prompt x

$$\begin{aligned} L_{CE}(x, o_w, o_l) &= -\log P(o_w \succ o_l | x) \\ &= -\log \sigma(r(x, o_w) - r(x, o_l)) \end{aligned}$$

Loss over the preference training set D is given by the following expectation

$$L_{CE} = -\mathbb{E}_{(x, o_w, o_l) \sim \mathcal{D}} [\log \sigma(r(x, o_w) - r(x, o_l))]$$

RLHF: Learning the reward model $r(x, o)$

$$L_{CE} = -\mathbb{E}_{(x, o_w, o_l) \sim \mathcal{D}} [\log \sigma(r(x, o_w) - r(x, o_l))]$$

- To learn a reward model using this loss function, we can use any regression model that can take text as input and generate a scalar output
- For example:
 - Initialize a reward model from a pretrained LLM
 - To generate scalar outputs, we replace the language modeling head with a single dense linear layer
 - Then gradient descent with the above loss function to learn to score model output using the preference data

RLHF: Now we have the reward model...

Now that we have a reward model, we need a policy that optimizes the reward!

The NLP version of RL is different from the traditional RL:

- In NLP, we are not starting from scratch to learn the optimal policy. We want to guide the pretrained LLM towards our preferred behaviors.
 - Given the small amount of preference data, LLM will forget everything to maximize reward
 - **there should be penalties for drifting too far from the original pretrained behavior**

RLHF: Learning a **policy** that optimizes the reward

The objective we want to maximize

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, o \sim \pi_{\theta}(o|x)} [r(x, o) - \beta \mathbb{D}_{\text{KL}}[\pi_{\theta}(o|x) || \pi_{\text{ref}}(o|x)]]$$

- KL divergence measure the distance between 2 probability distributions.
- The β term is a hyperparameter that modulates the impact of this penalty term
- For LLM-based policies, the KL divergence is the log is the ratio of the trained policy π_{θ} to the original reference policy π_{ref} , so we can rewrite

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, o \sim \pi_{\theta}(o|x)} \left[r_{\phi}(x, o) - \beta \frac{\pi_{\theta}(o|x)}{\pi_{\text{ref}}(o|x)} \right]$$

Learning a **policy** that optimizes the reward

$$\pi^* = \underset{\pi_\theta}{\operatorname{argmax}} \mathbb{E}_{x \sim \mathcal{D}, o \sim \pi_\theta(o|x)} \left[\underbrace{r_\phi(x, o)}_{\text{Want high reward}} - \beta \underbrace{\frac{\pi_\theta(o|x)}{\pi_{\text{ref}}(o|x)}}_{\text{Penalty with KL: staying close to the original model}} \right]$$

Sample from policy

Want high reward

Penalty with KL: staying close to the original model

Direct Preference Optimization (DPO)

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov^{*†}

Archit Sharma^{*†}

Eric Mitchell^{*†}

Stefano Ermon^{†‡}

Christopher D. Manning[†]

Chelsea Finn[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

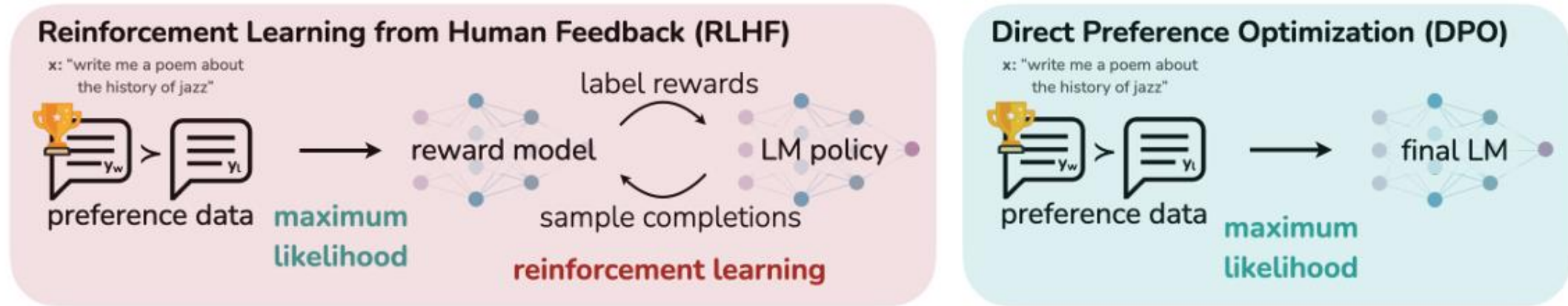
Direct Preference Optimization (DPO)

- Avoid RL all together!

“DPO directly optimizes for the policy best satisfying the preferences with *a simple classification objective*,

fitting an implicit reward model whose corresponding optimal policy can be extracted in closed form.” (Rafailov et al., 2023)

Direct Preference Optimization (DPO)



$$L_{\text{DPO}}(\pi_{\theta}) = -\mathbb{E}_{(x, o_w, o_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(o_w|x)}{\pi_{\text{ref}}(o_w|x)} - \beta \log \frac{\pi_{\theta}(o_l|x)}{\pi_{\text{ref}}(o_l|x)} \right) \right]$$

Where o_w is preferred output, o_l is dispreferred output

Direct Preference Optimization (DPO)

- DPO doesn't learn an explicit reward model like Bradley-Terry model

$$\begin{aligned}P(o_i \succ o_j | x) &= \sigma(z_i - z_j) \\ &= \sigma(r(x, o_i) - r(x, o_j))\end{aligned}$$

- DPO rewrite the closed-form solution to this maximization to express the reward function $r(x, o)$ in terms of optimal policy π^* and reference policy π_{ref}

$$r(x, o) = \beta \log \frac{\pi_r(o|x)}{\pi_{ref}(o|x)} + \beta \log Z(x)$$

Direct Preference Optimization (DPO)

- DPO *rewrite* the closed-form solution to this maximization to express the reward function $r(x,o)$ in terms of optimal policy π^* and reference policy π_{ref}
 - Closed-form: an equation that you can optimize directly without running RL

$$r(x,o) = \beta \log \frac{\pi_r(o|x)}{\pi_{ref}(o|x)} + \beta \log Z(x)$$

$Z(x)$ is a partition function: a sum over all the possible outputs o given a prompt x

$$Z(x) = \sum_y \pi_{ref}(o|x) \exp \left(\frac{1}{\beta} r(x,o) \right)$$

Direct Preference Optimization (DPO)

- DPO expresses the likelihood of a preference pair in terms of the trainable model and the base (reference) model: $\pi_\theta \pi_{ref}$, rather than in terms of an explicit reward model

$$L_{DPO}(x, o_w, o_l) = -\log \sigma \left(\beta \log \frac{\pi_\theta(o_w|x)}{\pi_{ref}(o_w|x)} - \beta \log \frac{\pi_\theta(o_l|x)}{\pi_{ref}(o_l|x)} \right)$$

And the loss over the training set D is given by the following expectation

$$L_{DPO}(\pi_\theta) = -\mathbb{E}_{(x, o_w, o_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(o_w|x)}{\pi_{ref}(o_w|x)} - \beta \log \frac{\pi_\theta(o_l|x)}{\pi_{ref}(o_l|x)} \right) \right]$$

Direct Preference Optimization: Putting it together

A loss function on
reward functions



A transformation
between reward
functions and policies



A loss function
on policies

Derived from the Bradley-Terry model of human preferences:

$$\mathcal{L}_R(r, \mathcal{D}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log \sigma(r(x, y_w) - r(x, y_l))]$$

$$r_{\pi_\theta}(x, y) = \beta \log \frac{\pi_\theta(y | x)}{\pi_{\text{ref}}(y | x)} + \beta \log Z(x)$$

When substituting, the $\log Z$ term cancels, because the loss only cares about **difference** in rewards

Reward of
preferred
response

Reward of
dispreferred
response

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

Advantages of DPO

- DPO does not require training an explicit reward model.
- DPO learns directly from the preference contained in the preference dataset without the need for expensive online sampling from π_θ
- More stable
- Implementation complexity is relatively low

Models Trained With DPO

huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Open LLM Leaderboard

The Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.

Submit a model for automated evaluation on the GPU cluster on the "Submit" page! The leaderboard's backend runs the great [Eleuther AI LLaMArena Model Evaluation framework](#) - read more details in the "About" page!

LLM Benchmark Metrics through time About Submit here!

Search for your model (separate multiple queries with "," and press ENTER)

Select columns to show

☒ Average ☒ ARC ☒ HellaSwag ☒ MMLU ☒ TruthfulQA ☒ Winogrande ☒ GSM8K ☐ Type ☐ Architecture ☐ Precision ☐ Merged ☐ Hub License ☐ #Params (B) ☐ Hub ☐ Model sha

☐ Show private/delisted models ☐ Show merges ☐ Show MDE ☐ Show flagged models

Model types

☒ pre-trained ☒ fine-tuned ☒ instruction-tuned ☒ RL-tuned ☒ ?

Precision

☒ float32 ☒ float16 ☒ bfloat16 ☒ int8 ☒ int4 ☒ GPTQ ☒ ?

Model sizes (in billions of parameters)

☐ <7 ☐ 7-15 ☐ 15-30 ☒ 30-70 ☐ 70-130 ☐ 130-250 ☐ 250-500 ☐ 500-1000

T	Model	Average	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	GSM8K
1	yakari/finch	74.66	73.38	88.56	64.52	67.11	86.66	67.7
2	sliggit/UMA-TheBaggie-7B-v1	73.87	73.64	88	63.48	69.85	82.16	66.72
3	argilla/distilabelled-Marcosid-7B-sless	73.63	70.73	87.47	65.22	65.1	82.88	71.19
4	mlabonne/NeuralMarcosid-7B	73.57	71.42	87.59	64.84	65.64	81.22	79.74
5	abideen/NeoMixBus-7B	73.5	70.82	87.86	64.69	62.43	84.85	79.36
6	NeuroNexus/neuronex-7B-v0.1	73.44	73.04	88.32	61.15	75.82	86.66	62.47
7	argilla/distilabelled-Marcosid-7B-sless-hull	73.4	70.65	87.55	65.33	64.21	82	79.66
8	Galtrix/MistralTrix-v1	73.39	72.27	88.33	65.24	70.73	89.18	62.77
9	Isaadi/MesingCaterpillar	73.33	72.53	88.34	65.26	70.93	89.66	62.24
10	NeuroNexus/neuronex-7B-v0.3	73.29	72.7	88.26	65.1	71.35	89.9	61.41
11	Galtrix/MistralTrixfest	73.17	72.53	88.4	65.22	70.77	81.37	68.73
12	sanix-fama/SanixDP1-v1	73.11	69.54	87.04	65.3	63.37	81.69	71.72
13	SentiMitsuki/Lelantos-DPO-7B	73.09	71.88	87.22	64	67.77	80.83	60.46

DPO
DPO (& VNA)
DPO
DPO
Merge (of DPO models)
DPO
DPO
DPO
DPO
DPO
No info but prob DPO, given Merge (incl. DPO)
DPO