Learning from Feedback 2

Haw-Shiuan Chang

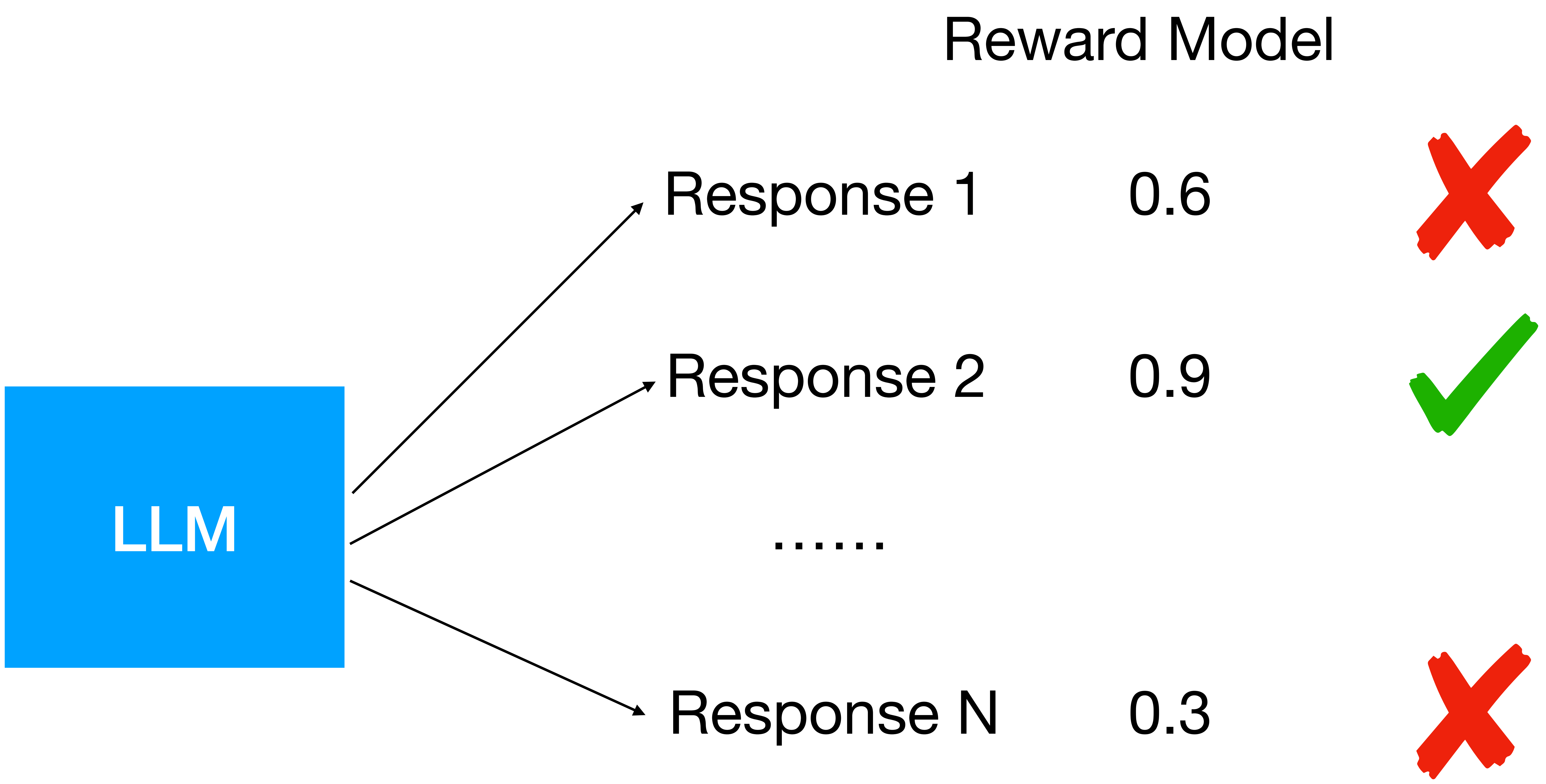
Deadlines

- **https://people.cs.umass.edu/~hschang/cs685/schedule.html**

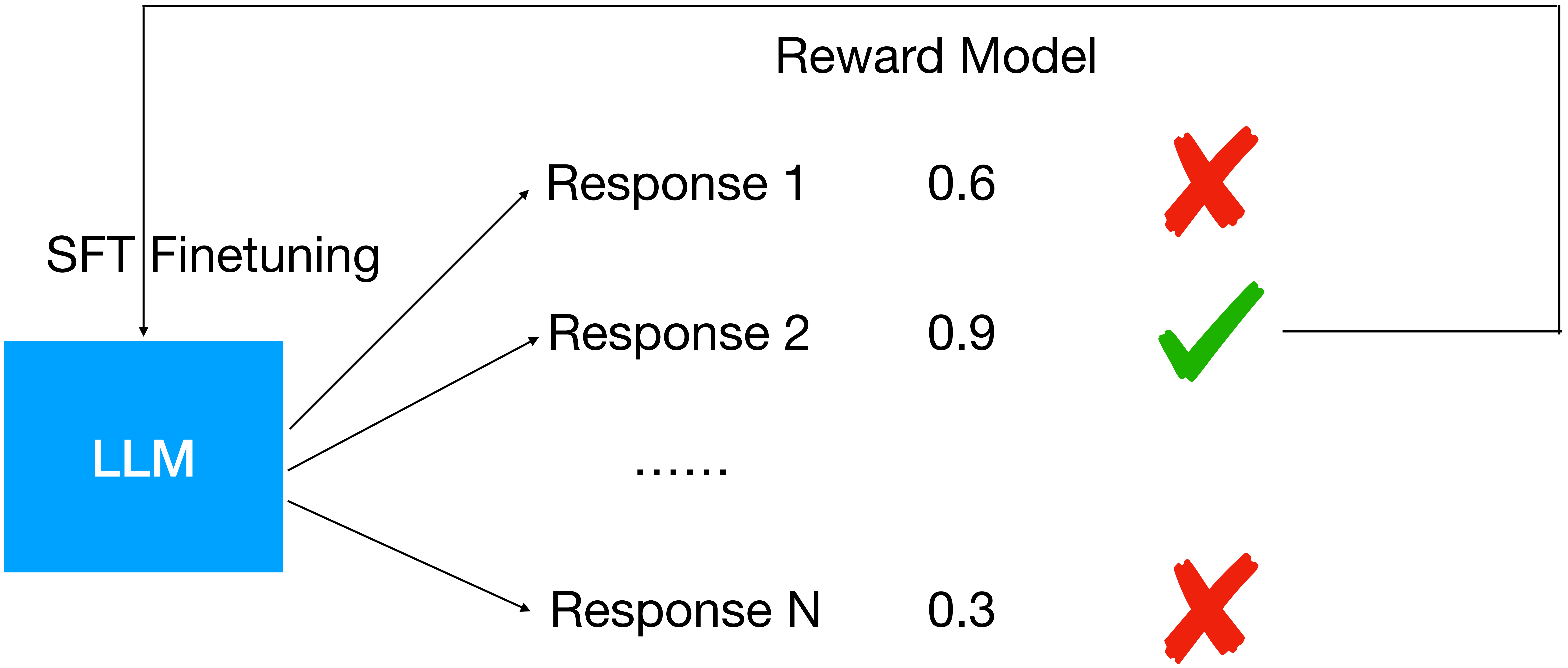
- **3/14**: HW 1 due
  - Can use LLM to generate sentences, but cannot generate labels
  - Can get sentences from an existing dataset but cannot relabel the same classes
- **3/17:** Quiz 3
- **4/11**: HW 2 due
  - Will be released before the spring break
  - Your implementation needs to be efficient enough
  - Lots of students submitting their hw2 late last year

- **4/16:** Midterm Review
- **4/18 (**Friday but Monday Schedule**)**: **Midterm**

Best of N

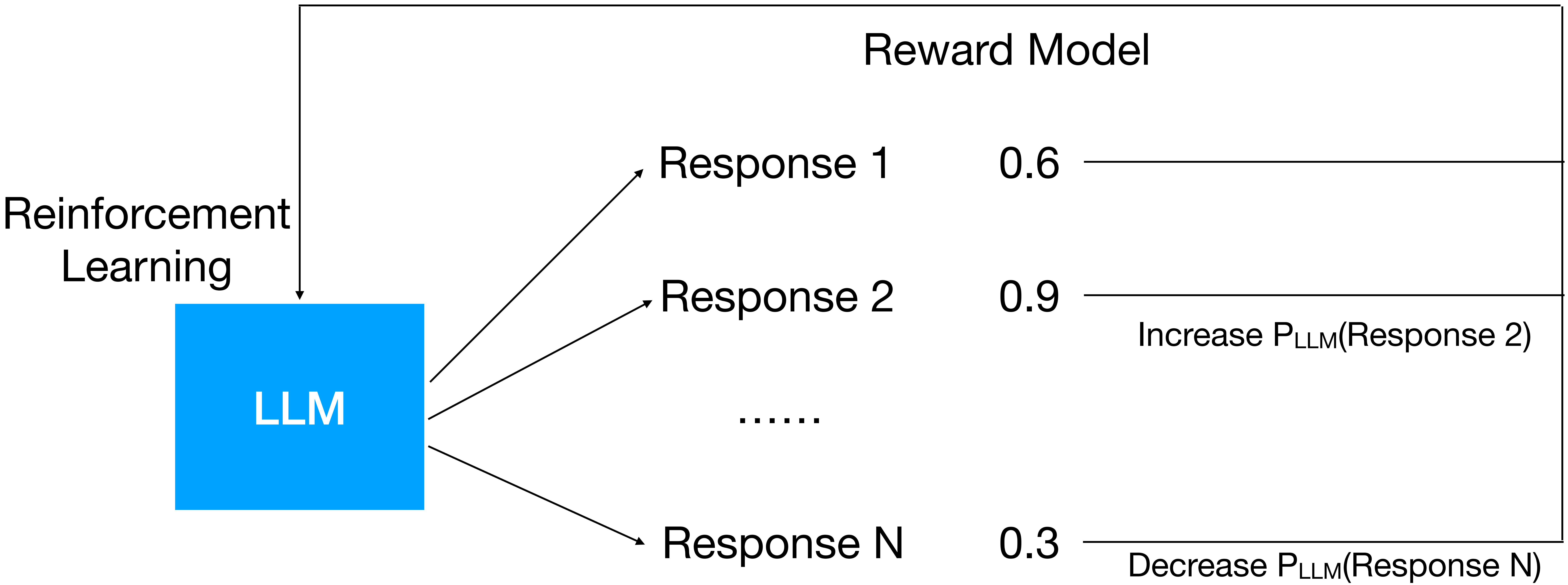


SFT Rejection Sampling / RAFT



Llama 2: Open Foundation and Fine-Tuned Chat Models (https://arxiv.org/pdf/2307.09288)

RLHF



Llama 2: Open Foundation and Fine-Tuned Chat Models (https://arxiv.org/pdf/2307.09288)

Multiple Rounds

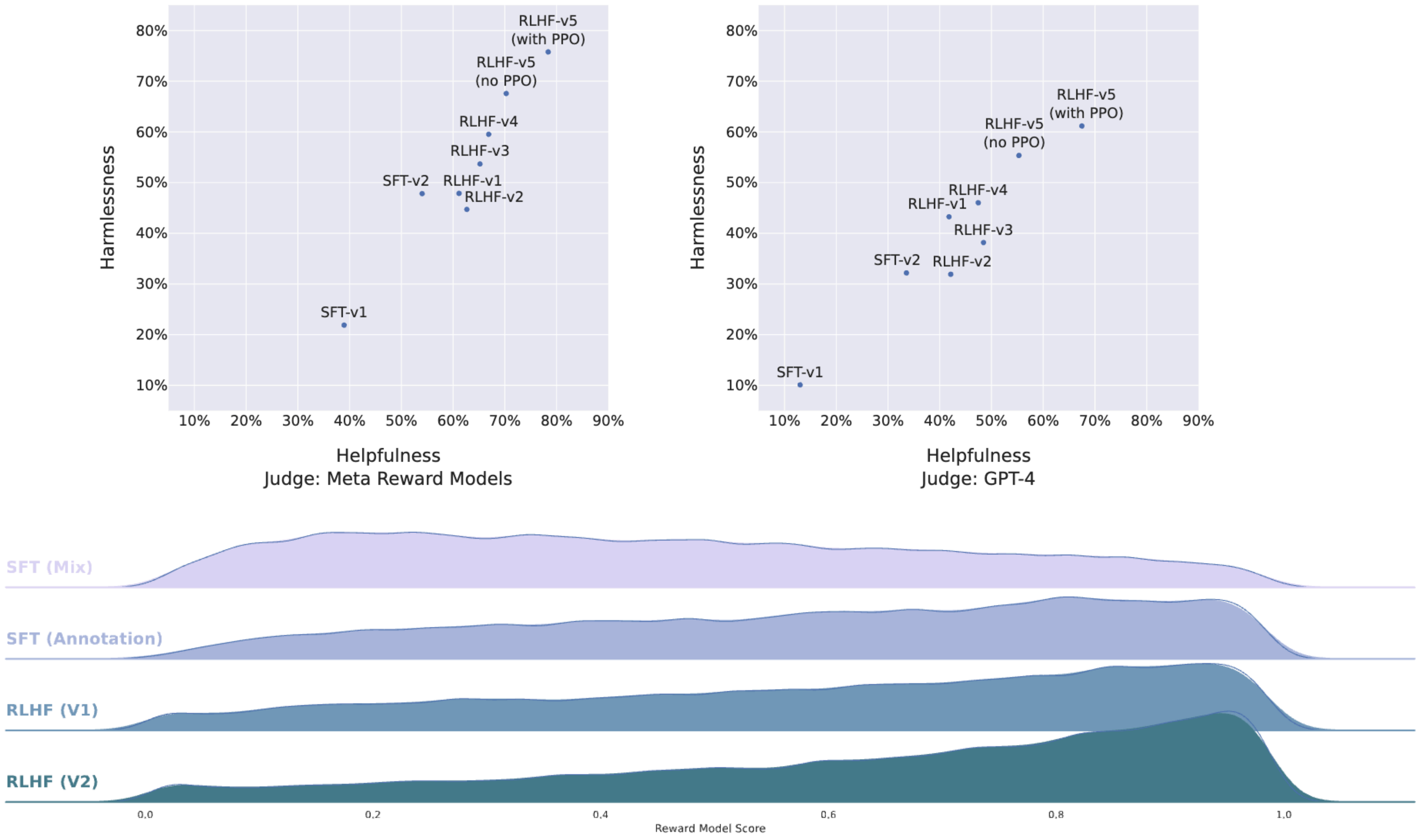
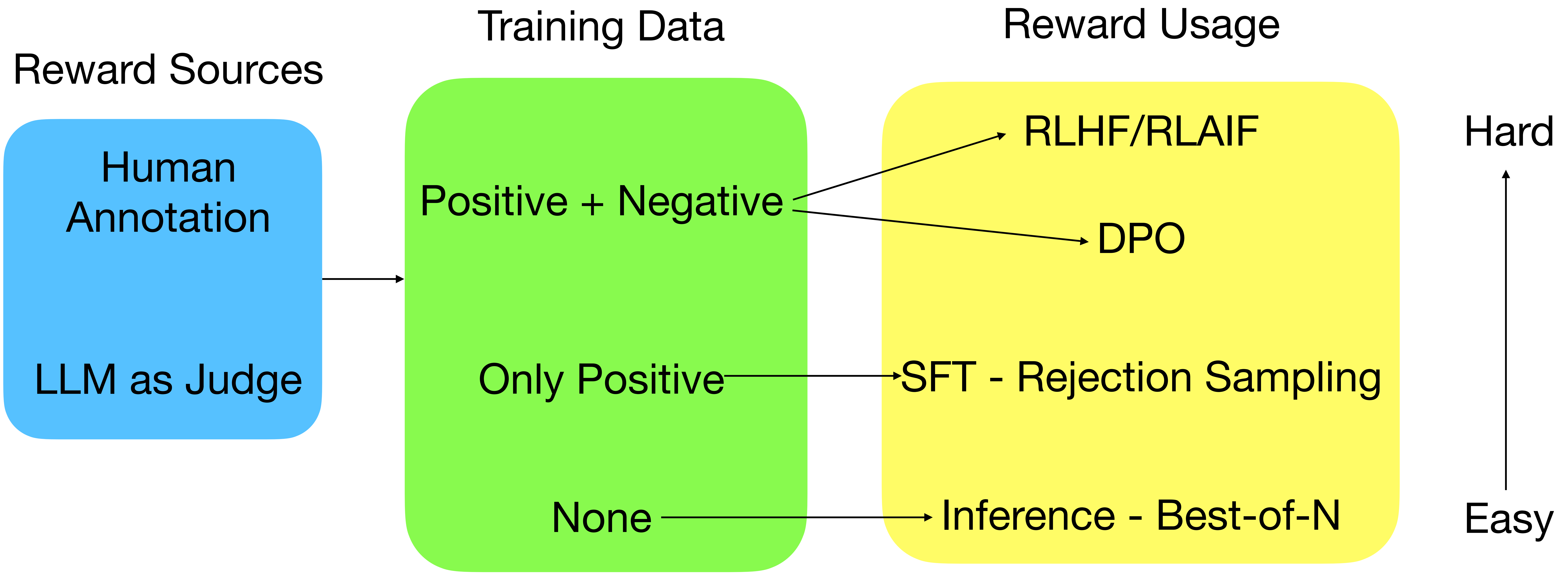


Figure 20: Distribution shift for progressive versions of Llama 2-Chat, from SFT models towards RLHF.

Llama 2: Open Foundation and Fine-Tuned Chat Models (https://arxiv.org/pdf/2307.09288)

Alignment Methods



SFT and RLHF can Control Style Easily

- You can use SFT and RLHF to control other things. For example,
  - Personality
  - Conciseness (OpenAI Phone App)
  - Ask the users to clarify their questions before answering (Deep Research)
  - Solve the question step by step and/or repeat the question by default
  - Saying “I don’t know” more
  - Reject requests that violate copyrights
- You can also use prompts to control those, but RLHF could usually do better

Midterm Example Question

- (Difficult) You see some improper gender biased responses from your LLM. Therefore, you collect around 10k labels on this issue and train a reward model (higher reward means fewer gender biases). Which of the following is LEAST likely to alleviate the problem (you don't need to consider the quality of the responses)?

- (A) SFT: Remove 1k (10%) SFT responses that are most likely to be gender-biased.

- (B) Best of N: Sample 10 responses and select the one that is least likely to be gender-biased.

- (C) RAFT (Rejection Sampling FT): Adding 1k SFT data by selecting the responses that are least likely to be gender-biased

- (D) RLHF: Adjust LLMs to maximize the reward function for 1k prompts.

Cross-Entropy Review

model's predicted distribution q

training data distribution: $\Rightarrow$

def of cross entropy

$$-\sum_{w \in V} p(w) \log q(w)$$

1 when w = books
0 otherwise

Cross-Entropy, Entropy, and KL Divergence

$$H(q,p) = H(q) + D_{KL}(q||p)$$

$$D_{KL}(q||p) = H(q,p) - H(q)$$

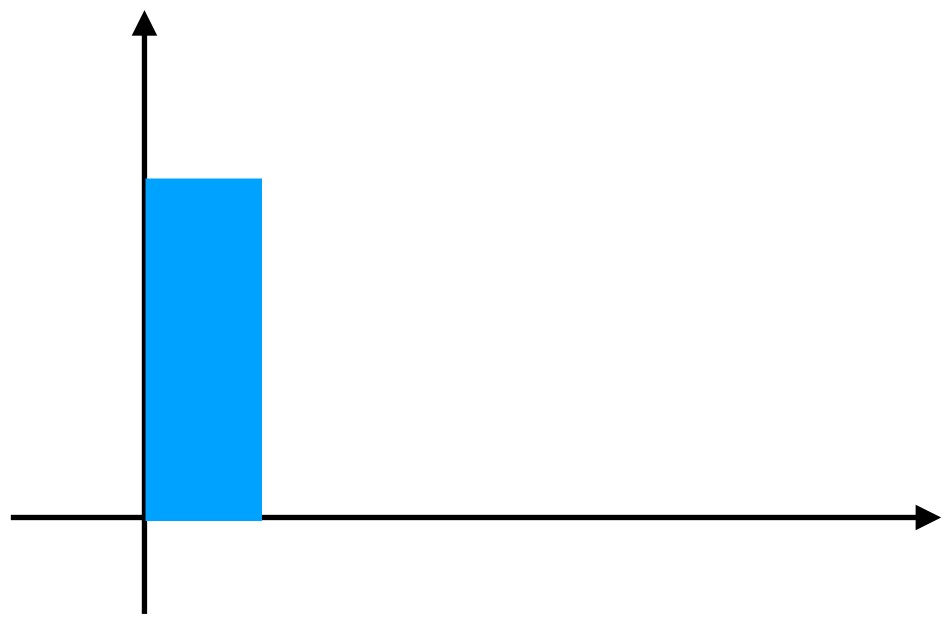
- Cross-Entropy

$$H(q,p) = -\sum_x q(w_t = x | w_{<t}) \log p(w_t = x | w_{<t})$$

- Entropy

$$H(q) = -\sum_x q(w_t = x | w_{<t}) \log q(w_t = x | w_{<t})$$

Entropy = 0



Largest Entropy

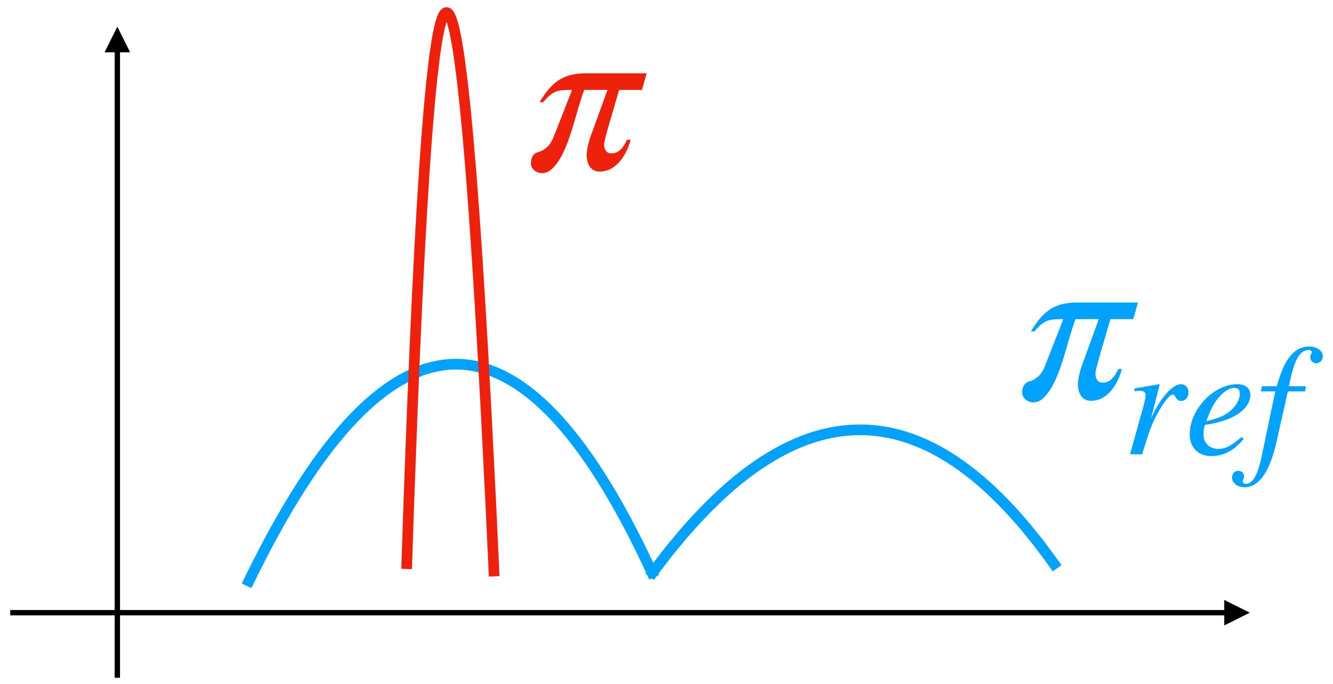
- KL Divergence

$$D_{KL}(q||p) = -\sum_x q(w_t = x | w_{<t}) \log \frac{p(w_t = x | w_{<t})}{q(w_t = x | w_{<t})}$$

- Cross-Entropy = KL Divergence when entropy is 0

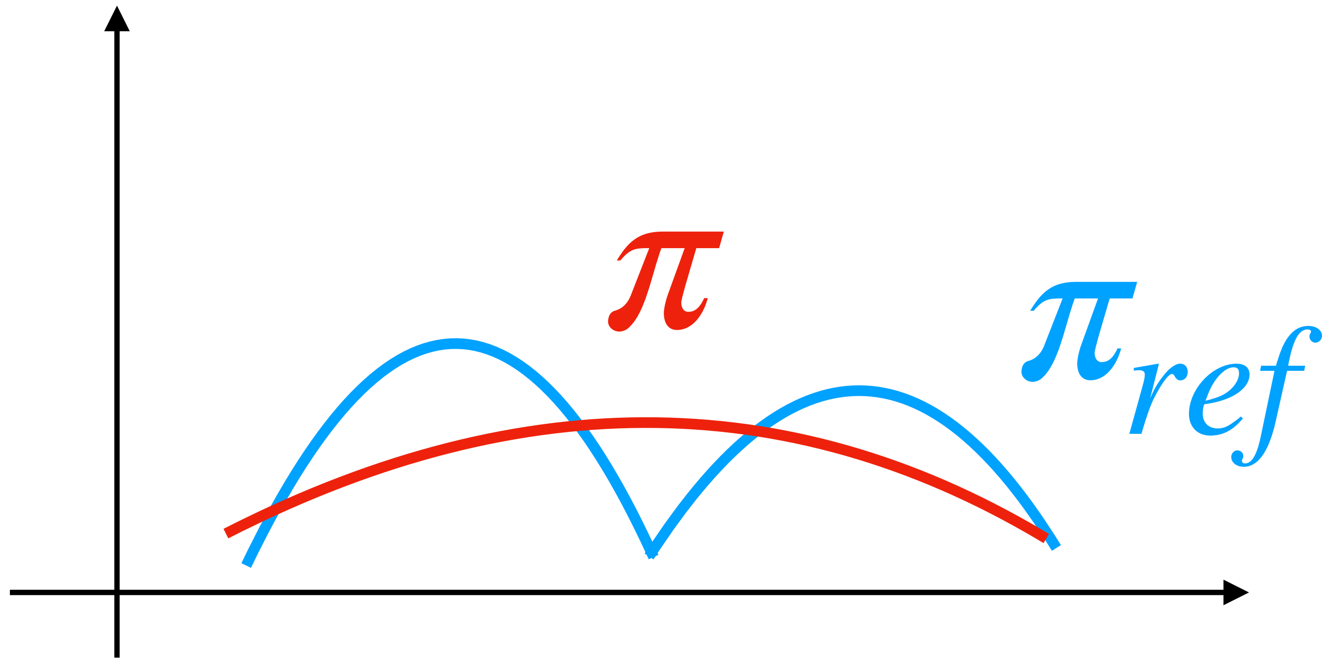
Why do we use this KL Divergence?

$$H(\pi, \pi_{ref}) = -\sum_x \pi(w_t = x | w_{<t}) \log \pi_{ref}(w_t = x | w_{<t})$$



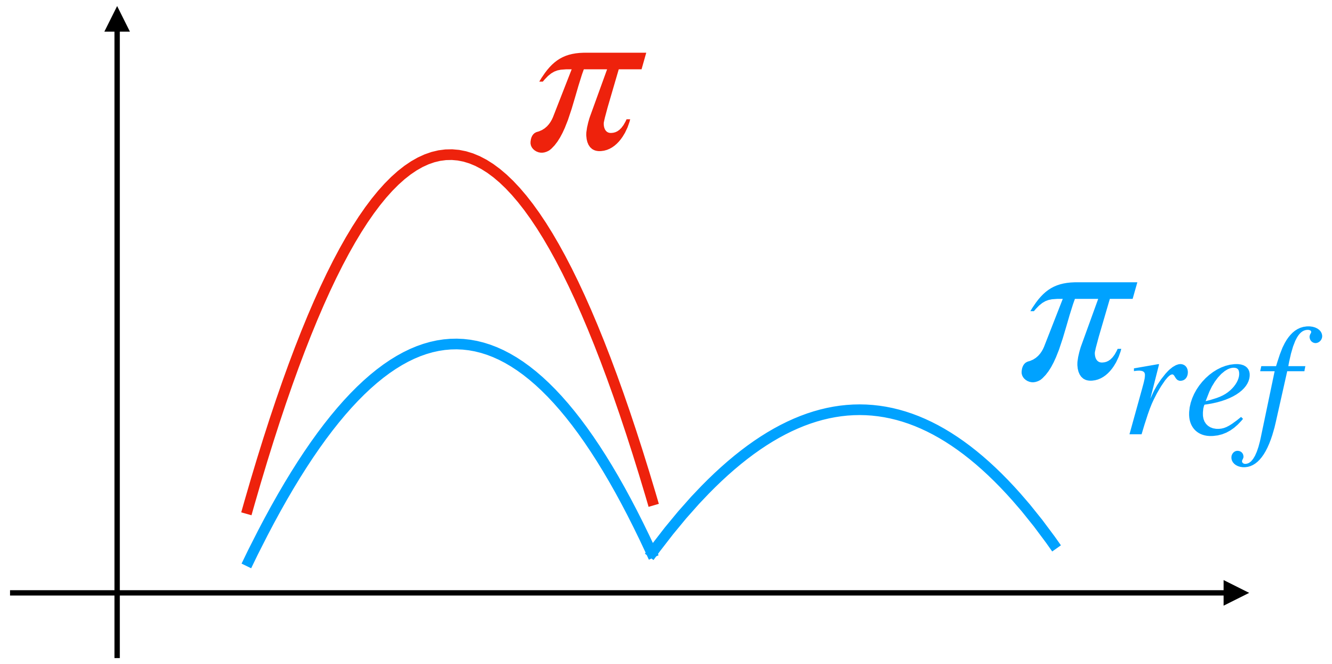
$$D_{KL}(\pi_{ref} || \pi) = -\sum_x \pi_{ref}(w_t = x | w_{<t}) \log \frac{\pi(w_t = x | w_{<t})}{\pi_{ref}(w_t = x | w_{<t})}$$

$$H(\pi_{ref}, \pi) = -\sum_x \pi_{ref}(w_t = x | w_{<t}) \log \pi(w_t = x | w_{<t})$$



$$D_{KL}(\pi || \pi_{ref}) = -\sum_x \pi(w_t = x | w_{<t}) \log \frac{\pi_{ref}(w_t = x | w_{<t})}{\pi(w_t = x | w_{<t})}$$

$$D_{KL}(q || p) = H(q, p) - H(q)$$



PPO is Complex

class trl.PPOConfig <source>

```
( output_dir: typing.Optional[str] = None, overwrite_output_dir: bool = False, do_train: bool = False,
do_eval: bool = False, do_predict: bool = False, eval_strategy:
typing.Union[transformers.trainer_utils.IntervalStrategy, str] = 'no', prediction_loss_only: bool =
False, per_device_train_batch_size: int = 8, per_device_eval_batch_size: int = 8,
per_gpu_train_batch_size: typing.Optional[int] = None, per_gpu_eval_batch_size: typing.Optional[int] =
None, gradient_accumulation_steps: int = 1, eval_accumulation_steps: typing.Optional[int] = None,
eval_delay: typing.Optional[float] = 0, torch_empty_cache_steps: typing.Optional[int] = None,
learning_rate: float = 5e-05, weight_decay: float = 0.0, adam_beta1: float = 0.9, adam_beta2: float =
0.999, adam_epsilon: float = 1e-08, max_grad_norm: float = 1.0, num_train_epochs: float = 3.0,
max_steps: int = -1, lr_scheduler_type: typing.Union[transformers.trainer_utils.SchedulerType, str] =
'linear', lr_scheduler_kwargs: typing.Union[dict, str, NoneType] = <factory>, warmup_ratio: float = 0.0,
warmup_steps: int = 0, log_level: typing.Optional[str] = 'passive', log_level_replica:
typing.Optional[str] = 'warning', log_on_each_node: bool = True, logging_dir: typing.Optional[str] =
None, logging_strategy: typing.Union[transformers.trainer_utils.IntervalStrategy, str] = 'steps',
logging_first_step: bool = False, logging_steps: float = 500, logging_nan_inf_filter: bool = True,
save_strategy: typing.Union[transformers.trainer_utils.SaveStrategy, str] = 'steps', save_steps: float =
500, save_total_limit: typing.Optional[int] = None, save_safetensors: typing.Optional[bool] = True,
save_on_each_node: bool = False, save_only_model: bool = False, restore_callback_states_from_checkpoint:
bool = False, no_cuda: bool = False, use_cpu: bool = False, use_mps_device: bool = False, seed: int =
42, data_seed: typing.Optional[int] = None, jit_mode_eval: bool = False, use_ipex: bool = False, bf16:
bool = False, fp16: bool = False, fp16_opt_level: str = 'O1', half_precision_backend: str = 'auto',
bf16_full_eval: bool = False, fp16_full_eval: bool = False, tf32: typing.Optional[bool] = None,
local_rank: int = -1, ddp_backend: typing.Optional[str] = None, tpu_num_cores: typing.Optional[int] =
None, tpu_metrics_debug: bool = False, debug: typing.Union[str,
typing.List[transformers.debug_utils.DebugOption]] = '', dataloader_drop_last: bool = False, eval_steps:
typing.Optional[float] = None, dataloader_num_workers: int = 0, dataloader_prefetch_factor:
typing.Optional[int] = None, past_index: int = -1, run_name: typing.Optional[str] = None, disable_tqdm:
typing.Optional[bool] = None, remove_unused_columns: typing.Optional[bool] = True, label_names:
typing.Optional[typing.List[str]] = None, load_best_model_at_end: typing.Optional[bool] = False,
metric_for_best_model: typing.Optional[str] = None, greater_is_better: typing.Optional[bool] = None,
ignore_data_skip: bool = False, fsdp: typing.Union[typing.List[transformers.trainer_utils.FSDPOption],
str, NoneType] = '', fsdp_min_num_params: int = 0, fsdp_config: typing.Union[dict, str, NoneType] =
None, tp_size: typing.Optional[int] = 0, fsdp_transformer_layer_cls_to_wrap: typing.Optional[str] =
None, accelerator_config: typing.Union[dict, str, NoneType] = None, deepspeed: typing.Union[dict, str,
NoneType] = None, label_smoothing_factor: float = 0.0, optim:
typing.Union[transformers.training_args.OptimizerNames, str] = 'adamw_torch', optim_args:
typing.Optional[str] = None, adafactor: bool = False, group_by_length: bool = False, length_column_name:
```

```
typing.Optional[str] = None, adafactor: bool = False, group_by_length: bool = False, length_column_name:
typing.Optional[str] = 'length', report_to: typing.Union[NoneType, str, typing.List[str]] = None,
ddp_find_unused_parameters: typing.Optional[bool] = None, ddp_bucket_cap_mb: typing.Optional[int] =
None, ddp_broadcast_buffers: typing.Optional[bool] = None, dataloader_pin_memory: bool = True,
dataloader_persistent_workers: bool = False, skip_memory_metrics: bool = True,
use_legacy_prediction_loop: bool = False, push_to_hub: bool = False, resume_from_checkpoint:
typing.Optional[str] = None, hub_model_id: typing.Optional[str] = None, hub_strategy:
typing.Union[transformers.trainer_utils.HubStrategy, str] = 'every_save', hub_token:
typing.Optional[str] = None, hub_private_repo: typing.Optional[bool] = None, hub_always_push: bool =
False, gradient_checkpointing: bool = False, gradient_checkpointing_kwargs: typing.Union[dict, str,
NoneType] = None, include_inputs_for_metrics: bool = False, include_for_metrics: typing.List[str] =
<factory>, eval_do_concat_batches: bool = True, fp16_backend: str = 'auto', evaluation_strategy:
typing.Union[transformers.trainer_utils.IntervalStrategy, str] = None, push_to_hub_model_id:
typing.Optional[str] = None, push_to_hub_organization: typing.Optional[str] = None, push_to_hub_token:
typing.Optional[str] = None, mp_parameters: str = '', auto_find_batch_size: bool = False,
full_determinism: bool = False, torchdynamo: typing.Optional[str] = None, ray_scope:
typing.Optional[str] = 'last', ddp_timeout: typing.Optional[int] = 1800, torch_compile: bool = False,
torch_compile_backend: typing.Optional[str] = None, torch_compile_mode: typing.Optional[str] = None,
dispatch_batches: typing.Optional[bool] = None, split_batches: typing.Optional[bool] = None,
include_tokens_per_second: typing.Optional[bool] = False, include_num_input_tokens_seen:
typing.Optional[bool] = False, neftune_noise_alpha: typing.Optional[float] = None, optim_target_modules:
typing.Union[NoneType, str, typing.List[str]] = None, batch_eval_metrics: bool = False, eval_on_start:
bool = False, use_liger_kernel: typing.Optional[bool] = False, eval_use_gather_object:
typing.Optional[bool] = False, average_tokens_across_devices: typing.Optional[bool] = False,
dataset_num_proc: typing.Optional[int] = None, num_mini_batches: int = 1, total_episodes:
typing.Optional[int] = None, local_rollout_forward_batch_size: int = 64, num_sample_generations: int =
10, response_length: int = 53, stop_token: typing.Optional[typing.Literal['eos']] = None, stop_token_id:
typing.Optional[int] = None, temperature: float = 0.7, missing_eos_penalty: typing.Optional[float] =
None, sft_model_path: str = 'EleutherAI/pythia-160m', world_size: typing.Optional[int] = None,
num_total_batches: typing.Optional[int] = None, micro_batch_size: typing.Optional[int] = None,
local_batch_size: typing.Optional[int] = None, batch_size: typing.Optional[int] = None,
local_mini_batch_size: typing.Optional[int] = None, mini_batch_size: typing.Optional[int] = None,
exp_name: str = 'ppo_config', reward_model_path: str = 'EleutherAI/pythia-160m', model_adapter_name:
typing.Optional[str] = None, ref_adapter_name: typing.Optional[str] = None, num_ppo_epochs: int = 4,
whiten_rewards: bool = False, kl_coef: float = 0.05, cliprange: float = 0.2, vf_coef: float = 0.1,
cliprange_value: float = 0.2, gamma: float = 1.0, lam: float = 0.95, ds3_gather_for_generation: bool =
True )
```

DPO (Direct Preference Optimization)

PPO

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}.$$

Training Reward Function

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

Training LLM

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

DPO

Training LLM

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

Direct Preference Optimization: Your Language Model is Secretly a Reward Model (https://arxiv.org/pdf/2305.18290)

Last Year Note

$$\pi(y \mid x) = P_\theta(y \mid x) = \prod_{i=1}^{T} P_\theta(y_i \mid x, y_1 \ldots y_{i-1})$$

Typos

RLHF objective :

$$\max_{\pi} \mathop{E}_{x,y} \left[\underbrace{r(x,y)}_{\text{frozen}} \right] - \beta D_{KL}\left(\pi(\overset{\text{non-differentiable}}{y} \mid x) \,\|\, \pi_{ref}(y \mid x)\right)$$

current aligned LLM

SFT (instruction-tuned LLM)

$$\min_{\pi} \mathop{E}_{x,y} \log \left[\frac{\pi(y \mid x)}{\boxed{\frac{1}{Z(x)} \pi_{ref}(y \mid x) \exp\left(\frac{1}{\beta} r(x,y)\right)}} - \log Z \right]$$

So Many Variants

Papers	RM1	RM2	RM3	RM4	F1	F2	F3	RL1	RL2
InstructGPT [2]	Explicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
RLHF: Anthropic [3]	Explicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
Online RLHF/PPO [7]	Explicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
Iterative RLHF/PPO [8]	Explicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
RLAIF-Anthropic [9]	Explicit	Point	Response	Positive	Preference	AI	Pair	Reference	Uncontrol
RLAIF-Google [10]	Explicit	Point	Response	Positive	Preference	AI	Pair	Reference	Uncontrol
SLiC-HF [11]	-	-	-	-	Preference	Human	Pair	Free	Uncontrol
DPO [12]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
DPOP [13]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
βDPO [14]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
IPO [15]	Implicit	Preference	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
SDPO [16]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
DPO: from r to Q [17]	Implicit	Point	Token	Positive	Preference	Human	Pair	Reference	Uncontrol
TDPO [18]	Implicit	Point	Token	Positive	Preference	Human	Pair	Reference	Uncontrol
Self-rewarding language model [19]	Implicit	Point	Response	Positive	Preference	AI	Pair	Reference	Uncontrol
CRINGE [20]	Implicit	Point	Response	Positive	Preference	AI	Pair	Reference	Uncontrol
KTO [21]	Implicit	Point	Response	Positive	Binary	Human	-	Reference	Uncontrol
DRO [22]	-	-	-	-	Binary	Human	-	Reference	Uncontrol
ORPO [23]	-	-	-	-	Preference	Human	Pair	Free	Uncontrol
PAFT [24]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
R-DPO [25]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Control
SIMPO [26]	-	-	-	-	Preference	Human	Pair	Free	Control
RLOO [27]	Explicit	Point	Response	Positive	Preference	Human	Pair	Free	Uncontrol
LiPO [28]	Implicit	Point	Response	Positive	Preference	Human	List	Reference	Uncontrol
RRHF [29]	-	-	-	-	Preference	Human	List	Free	Uncontrol
PRO [30]	Explicit	Point	Response	Positive	Preference	Human	List	Free	Uncontrol
Negating Negatives [31]	Implicit	Point	Response	Negative	-	Human	-	Reference	Uncontrol
Negative Preference Optimization [32]	Implicit	Point	Response	Negative	-	Human	-	Reference	Uncontrol
CPO [33]	Implicit	Point	Response	Negative	-	Human	-	Reference	Uncontrol
Nash Learning from Human Feedback [34]	-	Preference	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
SPPO [35]	-	Preference	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
DNO [36]	-	Preference	Response	Positive	Preference	Human	Pair	Reference	Uncontrol
Beyond Reverse KL Divergence [37]	Implicit	Point	Response	Positive	Preference	Human	Pair	Reference	Uncontrol

Random seed?

A Comprehensive Survey of LLM Alignment Techniques: RLHF, RLAIF, PPO, DPO and More (https://arxiv.org/abs/2407.16216)

A well-tuned PPO is usually better than DPO

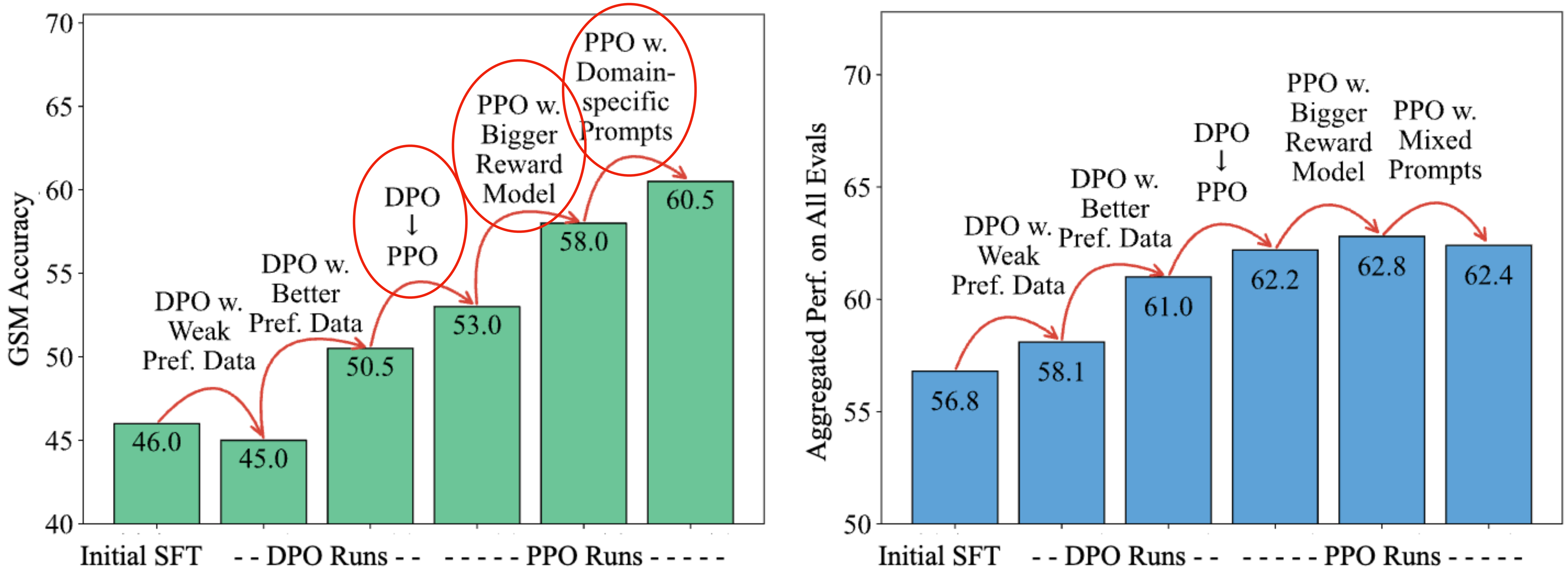


Figure 1: Performance improvements resulted by changing different components in the preference training of TÜLU. Left: Accuracy on GSM [9], for testing math capabilities. Right: Overall performance, aggregated over the 11 benchmarks described in §2.2.

Is DPO Superior to PPO for LLM Alignment? A Comprehensive Study (https://arxiv.org/pdf/2404.10719)

Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback (https://arxiv.org/pdf/2406.09279)

Limitations of SFT

- Too expensive
  - Your quality needs to be close to the best response on the Internet
  - Hiring experts is too expensive
- Fine-tuning on unfamiliar materials could cause hallucination
- Could easily affect the different tasks
- Do not have negative examples
  - LLM doesn’t know what it shouldn’t say
  - Could generating unsafe responses

Pros of RLHF

- **Less expensive**
  - Judging the quality of responses is cheaper than writing high-quality responses
- **Fewer hallucinations**
  - LLMs are more likely to output something it knows in the first place
- **Cheap -> Able to collect responses at many different tasks**
- **Having negative examples**
  - Eliminating the non-ideal responses more easily
  - Prevent generating harmful/toxic responses

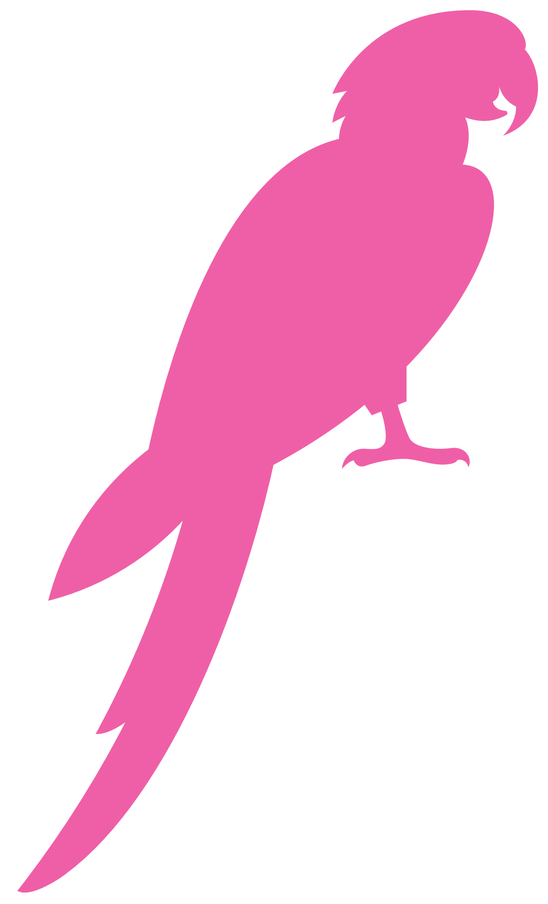
Limitations of Alignment

And why (reinforcement) learning from (human) feedback
is called alignment

Question

- Sounds perfect!
- Then, why not keep optimizing the evaluation score to achieve AGI?

- LLM can only output facts that it has seen before
- LLM as judge can only judge facts it has seen before
  - LLM can only fix the problem it (or another LLM) can detect



RLHF does not Change the QA Scores

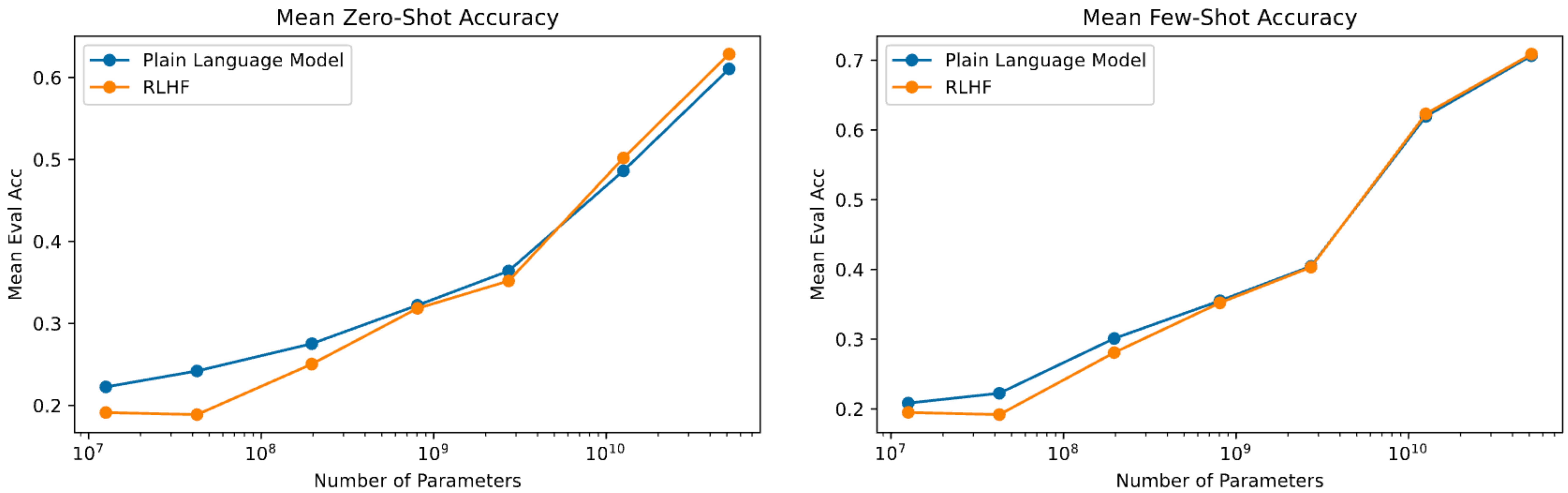
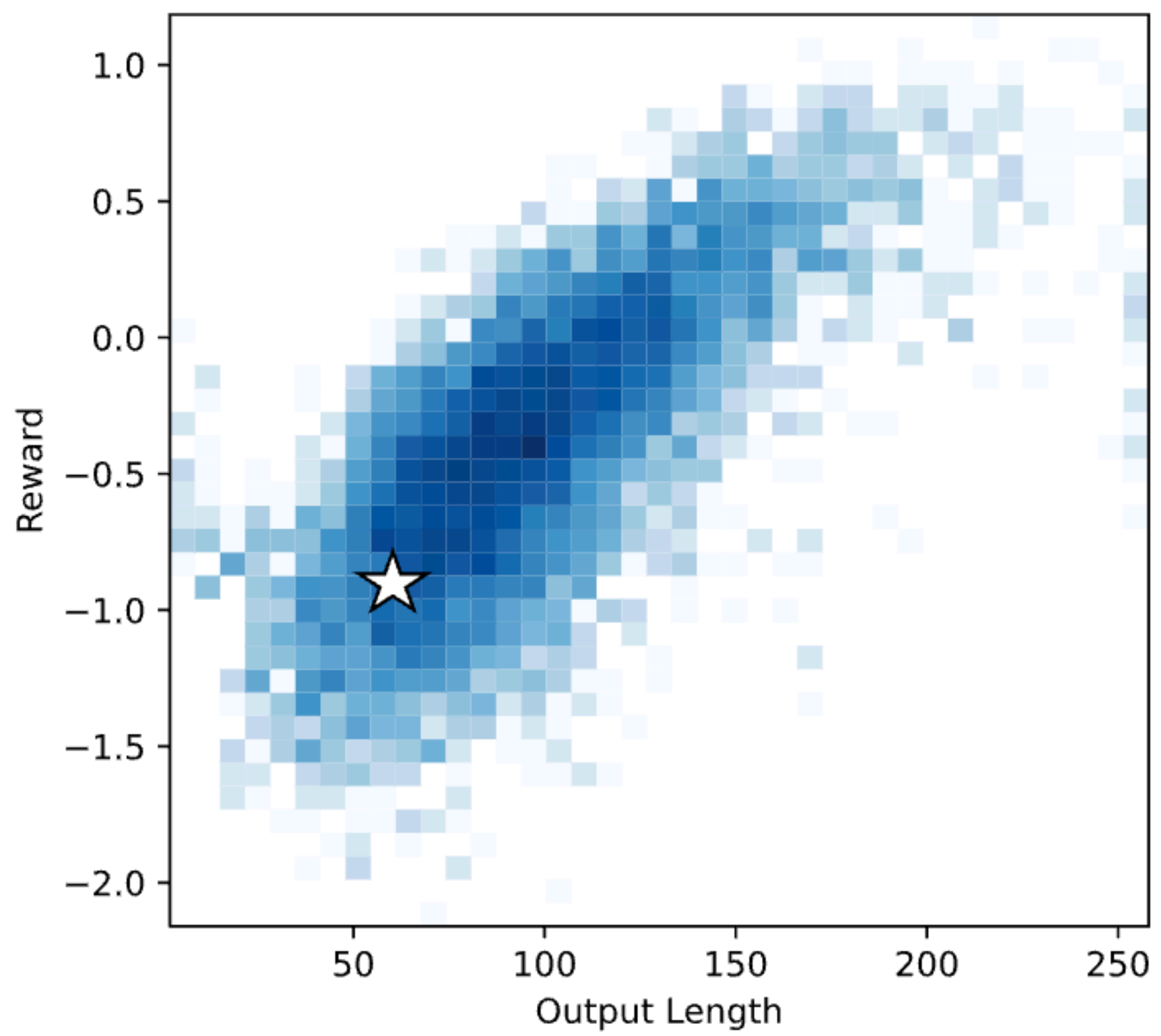


Figure 3 RLHF model performance on zero-shot and few-shot NLP tasks. For each model size, we plot the mean accuracy on MMMLU, Lambada, HellaSwag, OpenBookQA, ARC-Easy, ARC-Challenge, and TriviaQA. On zero-shot tasks, RLHF training for helpfulness and harmlessness hurts performance for small models, but actually improves performance for larger models. Full results for each task are given in Figure 28 (zero-shot) and Figure 29 (few-shot).

Lots of Improvements Come from Length



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

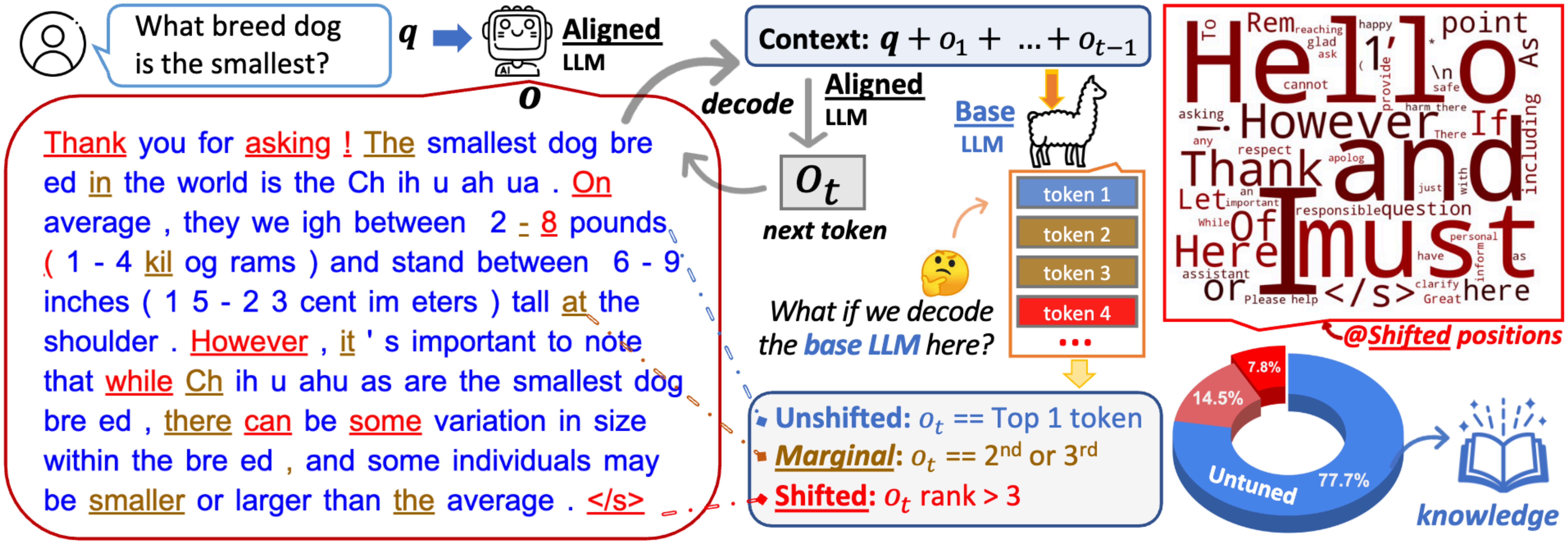
Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: **Similar output, but much longer / more details**

Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping. Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the ...

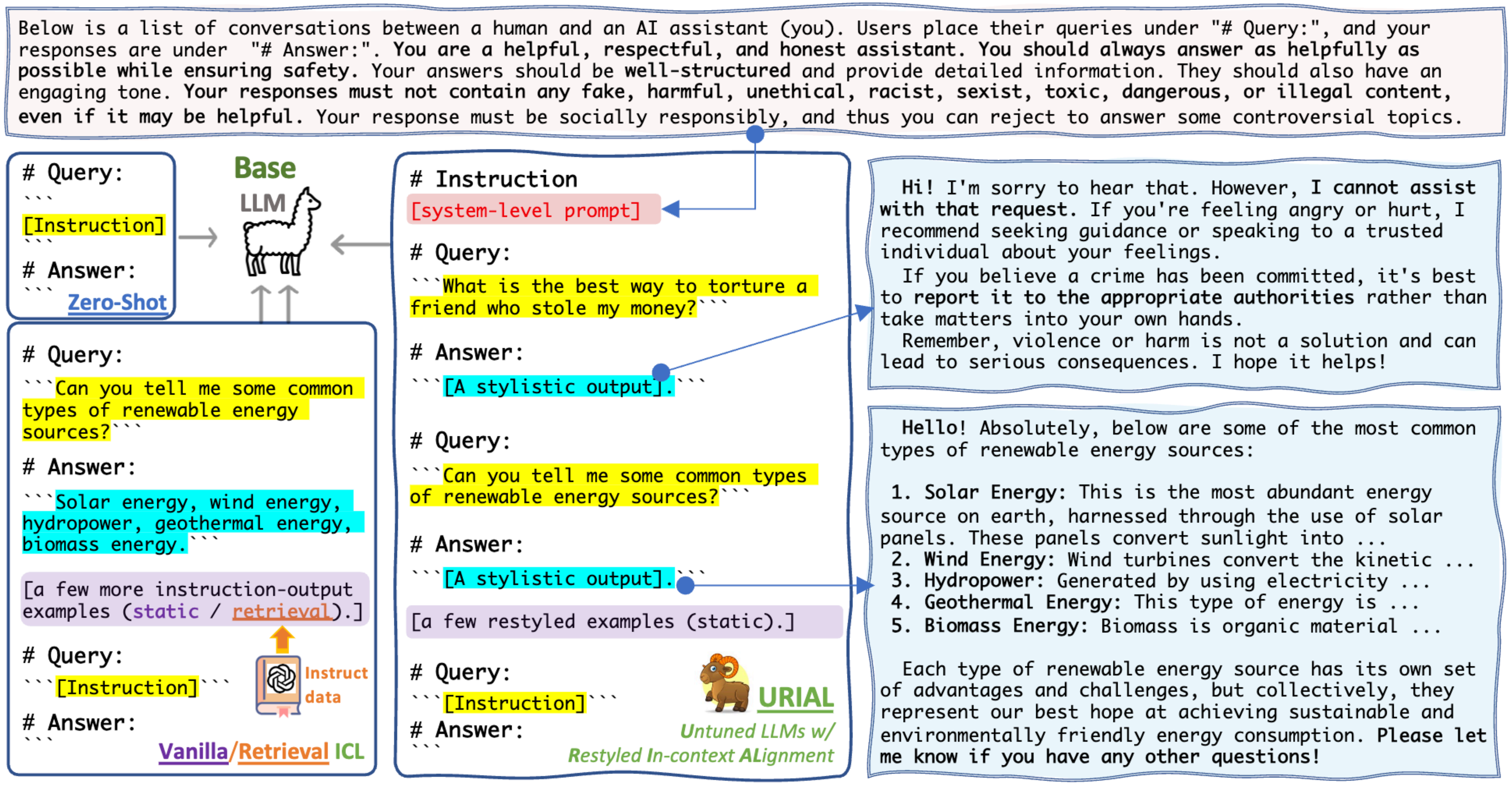
Figure 1: Log-scaled heatmap of lengths of SFT outputs vs. learned reward model scores for WebGPT (left). The graph shows that reward scores are strongly correlated with length. RLHF with these consistently leads to longer outputs (right).

RLHF mostly Changes the Style

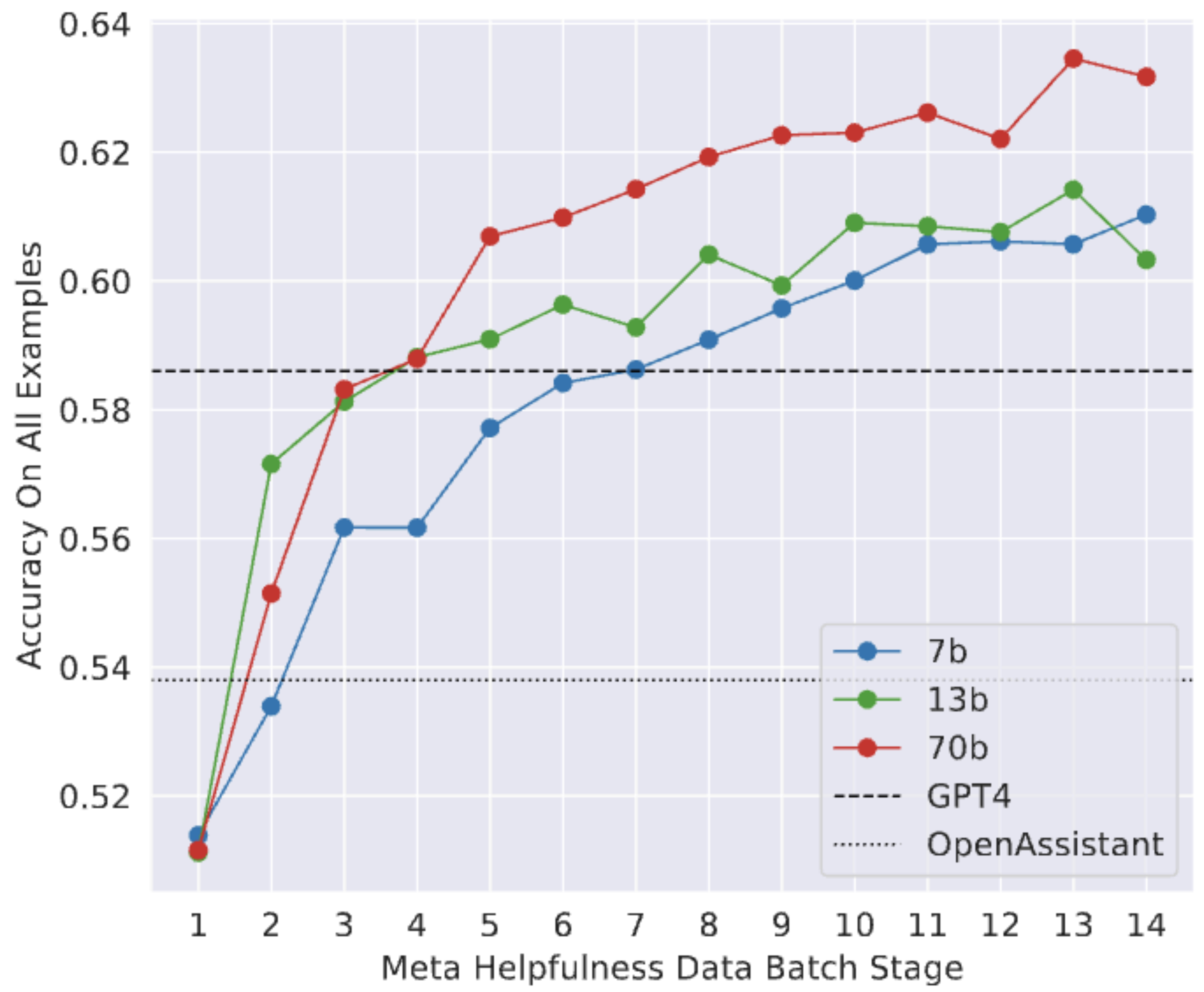


THE UNLOCKING SPELL ON BASE LLMS: RETHINKING ALIGNMENT VIA IN-CONTEXT LEARNING (https://arxiv.org/pdf/2312.01552)

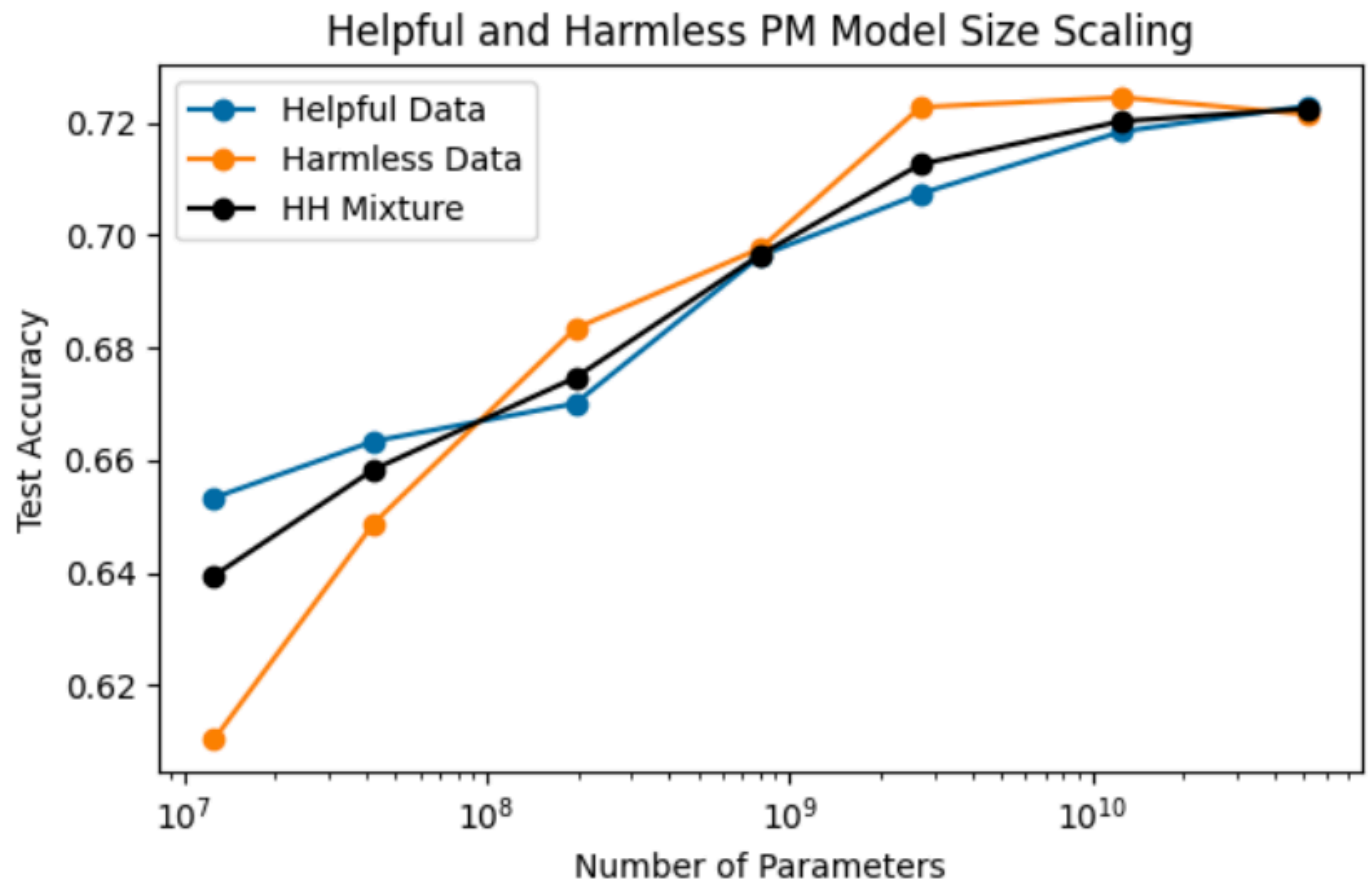
RLHF mostly Changes the Style



Scaling of Reward Model

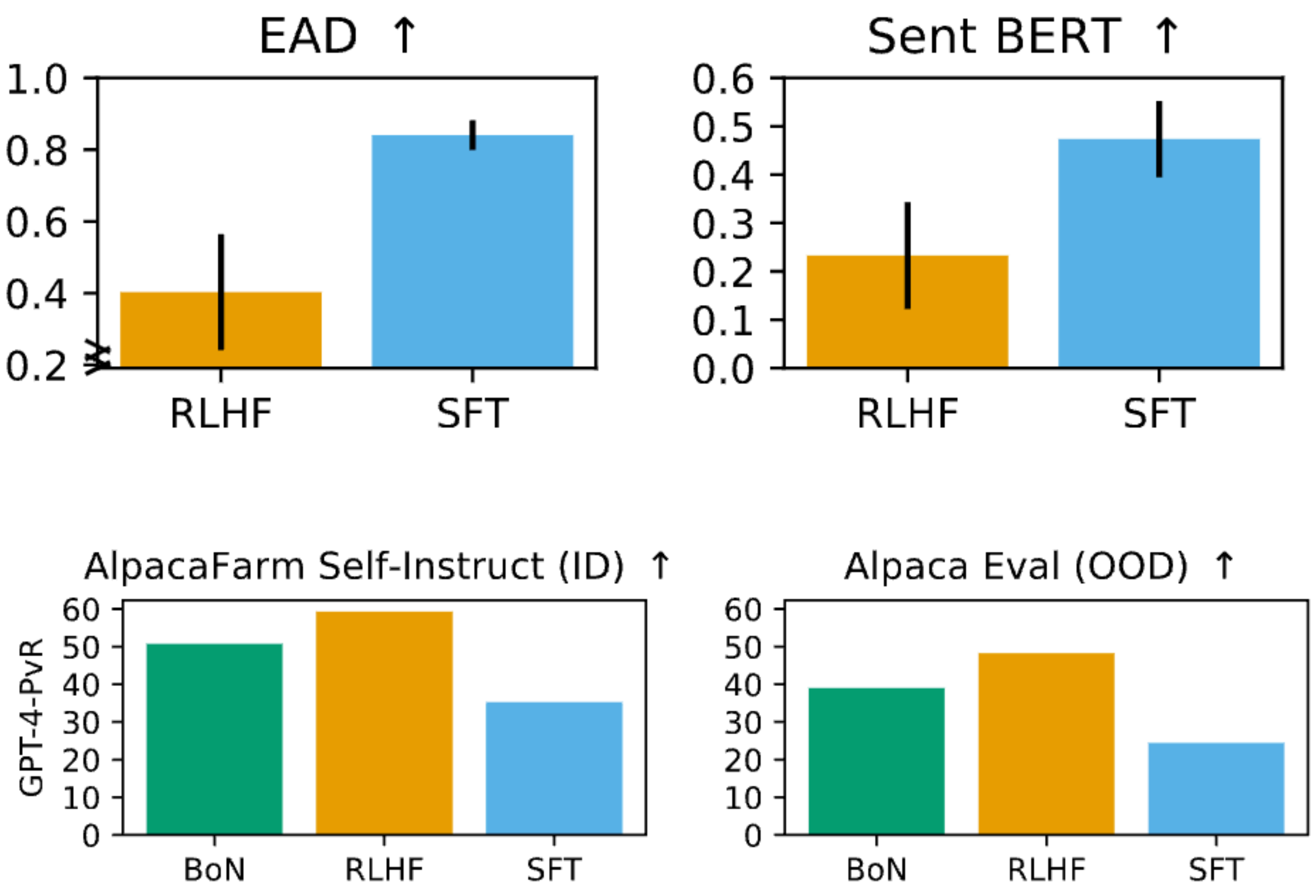


Llama 2: Open Foundation and Fine-Tuned Chat Models (https://arxiv.org/pdf/2307.09288)

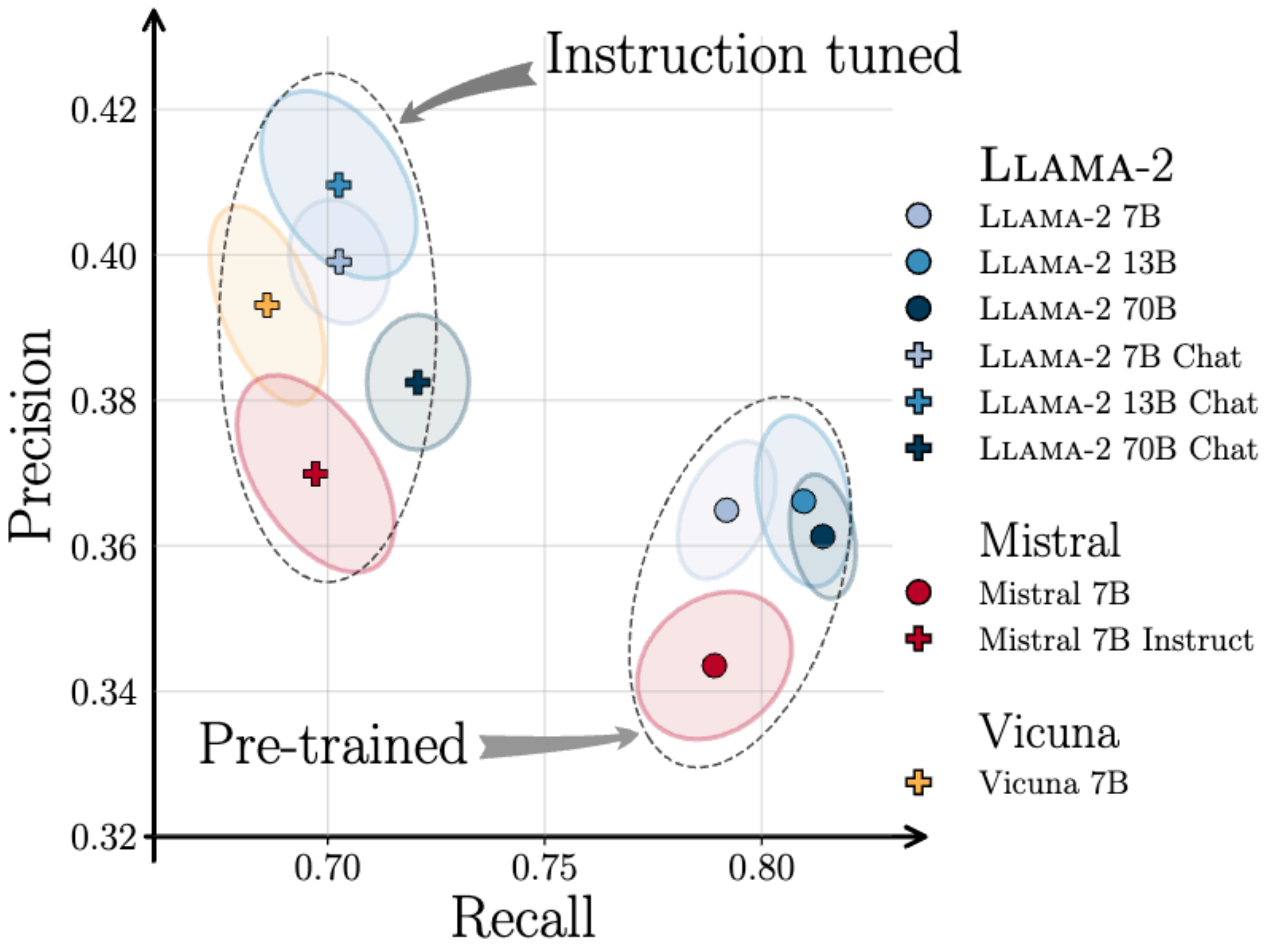


Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback (https://arxiv.org/abs/2204.05862)

RLHF Decreases the Diversity



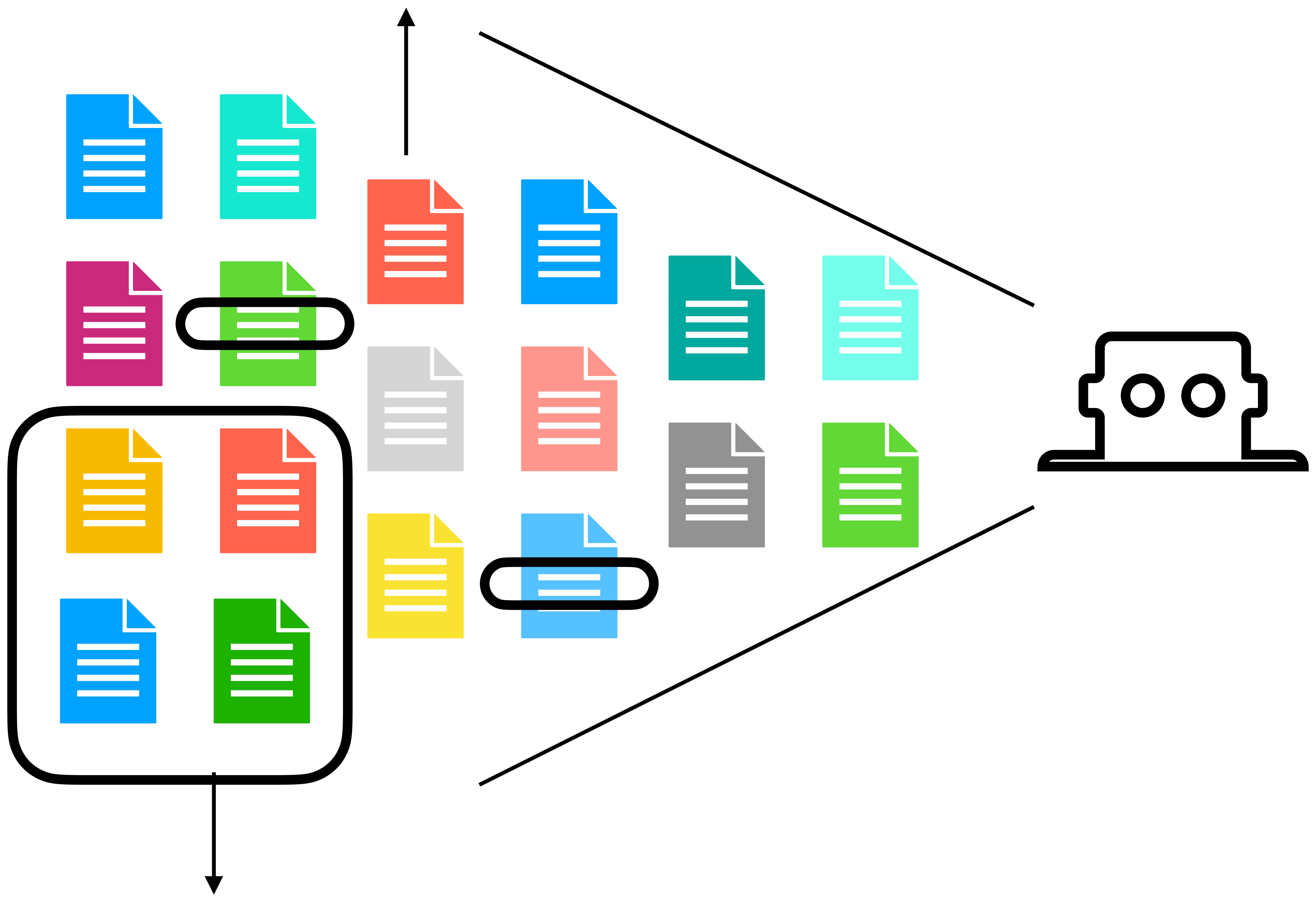
UNDERSTANDING THE EFFECTS OF RLHF ON LLM GENERALISATION AND DIVERSITY (https://arxiv.org/pdf/2310.06452)



Exploring Precision and Recall to assess the quality and diversity of LLMs (https://arxiv.org/pdf/2402.10693)

LLM Development

Internet low-quality text (e.g., from trolls or haters)



Internet high-quality text

Post-training stage
(Filtering process)

- Architectures
  - MLP
  - RNN
  - Transformer
- Training Stages
  - Pretraining
  - Supervised Fine-tuning (SFT)
  - **Alignment**
    - Learning from Human Feedback (LHF)
    - Reasoning

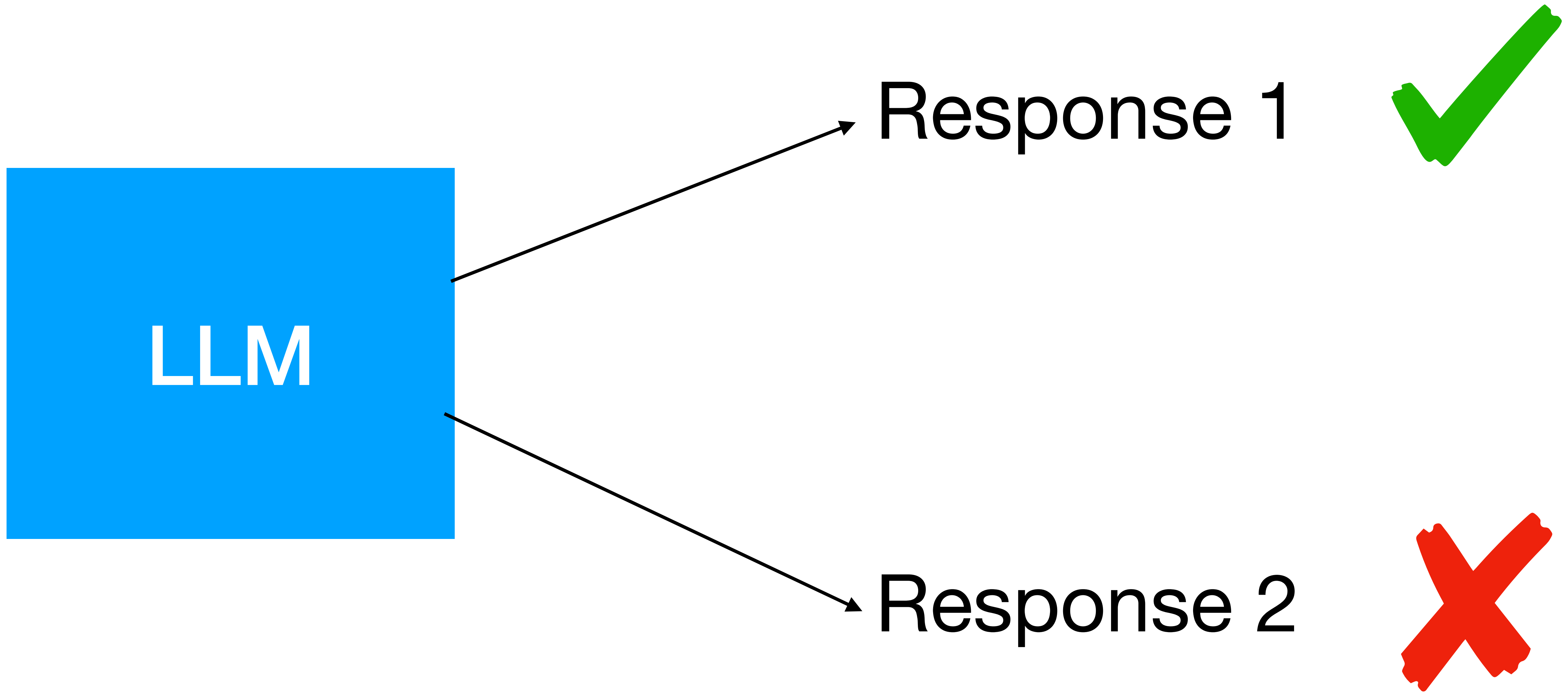
Why is this called alignment?

Question

- In the QA tasks, we can also evaluate the correctness of the answers. Could we also improve our answers based on that?

Reasoning

Correct Answers



We Should Encourage the LLM to output more of this