Instruction tuning / RLHP

- Goal: align LLMs with human prefs
  - make their outputs less harmful/toxic
  - increase relevance of outputs

- Two main methods:
  - supervised fine-tuning
  - reinforcement learning

---

**Instruction tuning:**

1. Collect a dataset of instructions on what task to solve, and outputs of that task for one or more examples.

Please answer the following question and provide a detailed justification.

What was the average of the CS685 S23 midterm?

I can’t answer that question b/c the midterm occurs on April 12 and it is March 27.

```
<table>
<thead>
<tr>
<th>LLM</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Fine-tune LLM to produce desired output given instruction
- Unlike what we've seen with 'pretrain+finetune', instruction tuning finetunes on many drift tasks at once.
- Instruction tuning improves generalization on tasks that are not seen during finetuning.
- Limitations:
  - Getting data is expensive, esp. for very complex tasks.
  - Some tasks don't have just one acceptable output.
  - Does not directly involve human prefs.

RLHF: reinforcement learning from human feedback

\[
\text{prefix X} \rightarrow \begin{bmatrix} S_1 \ & \ & \ S_2 \ & \ & \ S_3 \end{bmatrix} \quad \text{nucleus sampling}
\]

- Extremely expensive to obtain human feedback.
- Instead, we collect as many human judgments as we can, and then train a reward model to predict the human preferences.
  - Input: a prefix X, a sample S
  - Output: a scalar score, represents "overall quality" of the sample.
- assume we have two samples
  \[ S_{\text{good}} \gg S_{\text{bad}} \]
  \[ L_{RM} = \log \frac{R(S_{\text{good}})}{R(S_{\text{bad}})} \]

- Intuitively, the good sample's reward should be higher than that of the bad sample.

We can now use our reward model to obtain a score \( R(S) \) of any sample \( S \) generated from a prefix without needing a human labeler.

- Reward model is trained to mimic human prefs.
How do we use our reward model to better align LMs to human prefs?

1. Overgeneration + reranking ("best-of-n")
   - generate n samples, score each
   - w/ reward model, pick the one w/ highest reward
   - no further training required

2. just fine-tune the LM to maximize
   \[ P(S_{good} | x) \]
   - issue: what if \( S_{good} \) is not the only acceptable high-reward sample
   - what if \( S_{good} \) itself is bad

3. use reinforcement learning to increase
   \[ P(S_{good} | x) \]
   by a small amount, decrease
   \[ P(S_{bad} | x) \]
   by a small amount, where these amounts are functions of the rewards
   \[ R(S_{good}), R(S_{bad}) \]
RLHF:

- we observe a reward only after generating a full (multi-token) sample via a decoding algo

- goal: maximize \( p(s_{\text{good}}|x) \), minimize \( p(s_{\text{bad}}|x) \)
  subject to the rewards
  
  \[
  \text{RL Loss: } f(R(s), p(s|x))
  \]

  \( \downarrow \) REINFORCE (Williams 1992)

  \( \downarrow \) PPO (Schulman 2016)

  \( \downarrow \) used in ChatGPT/GPT4, etc

- important not to deviate too much from the base LM, to prevent reward hacking

- add another loss at the token level
  that approximates KL divergence between
  the current model \( P_{\text{RLHF}} \) and the original \( P_{\text{base}} \):

  for a given word \( w_i \)

  \[
  \log \frac{P_{\text{RLHF}}(w_i|w_{i-1},...,x)}{P_{\text{base}}(w_i|w_{i-1},...,x)}
  \]
- making it work in practice:

1. Initial model: LM
   - Supervised fine-tuning on high-quality examples
   - "Instruction tuning"

2. Instruction-tuned LM
   - Generates samples
   - Gets human judgment
   - Train with PPO using predicted rewards

3. RLHF-aligned model

4. Repeat process with humans

Humans