teacher forcing during SFT:

Complete this Python fn to return a list of even numbers up to n

```
def evens(n):
```

Ground-truth output (used in SFT):

```
results = []
for i in range(n):
    if i%2 == 0:
        results.append(i)
return results
```

at each time step model is fed correct prefix

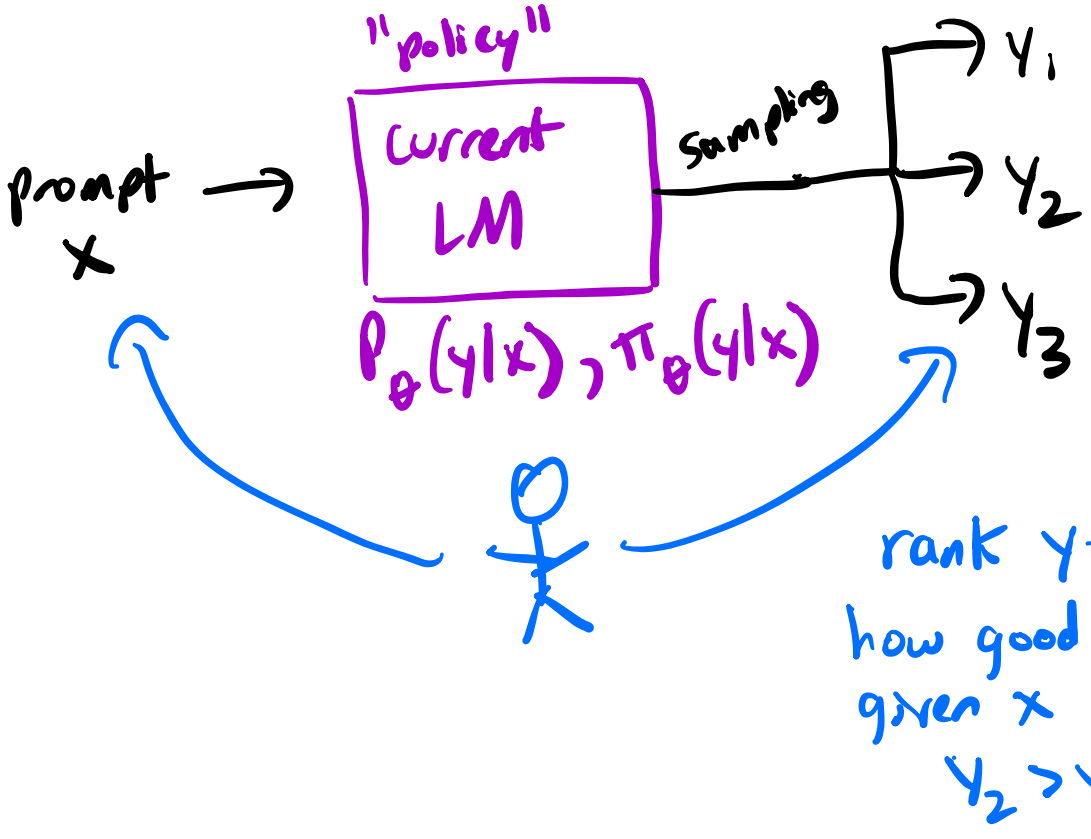
Model sample:

```
results = {}
for i in range(n):
    if i%2==0:
        results.append(i)
return results
```

error made during decoding

model never sees errors like these during SFT

improve our model w/o teacher forcing:



key idea: increase $p_\theta(y_w|x)$

↳ strongly preferred

decrease $p_\theta(y_L|x)$

↳ strongly dispreferred

↳ obv can't have human raters in the loop

↳ offline: collect a large dataset of human prefs

$x, y_w > y_L$

↳ can we train a model on this dataset to predict human prefs

↳ reward model

↳ input: prompt x_i, response y_i

↳ output: scalar score $r(x, y_i)$

↳ train on dataset of human prefs

↳ x, y_w, y_L

↳ Bradley-Terry pairwise pref. model

$$p(y_w > y_L \mid x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_L))}$$

loss $L = -\log\left(\nearrow\right)$

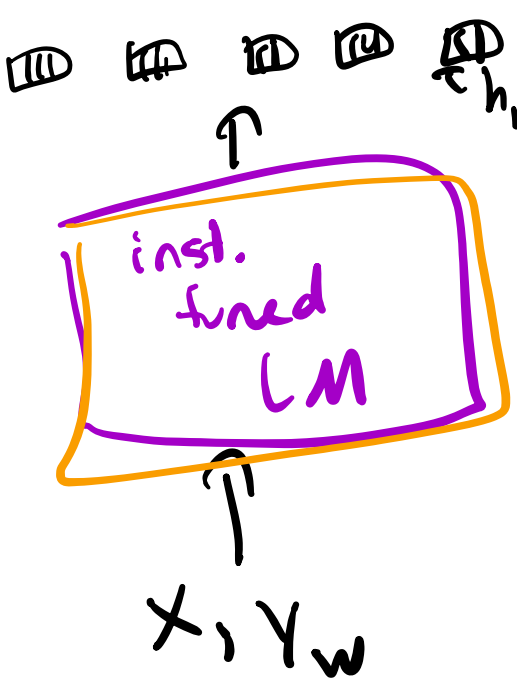
∴ simplifies

$$L = -\log \sigma\left(r(x, y_w) - r(x, y_L)\right)$$

↳ $\sigma(x) = \frac{1}{1+e^{-x}}$

intuition: to minimize loss, y_w should get higher reward than y_l

architecture: scalar linear layer on top of the final-layer last hidden state of our inst. tuned model

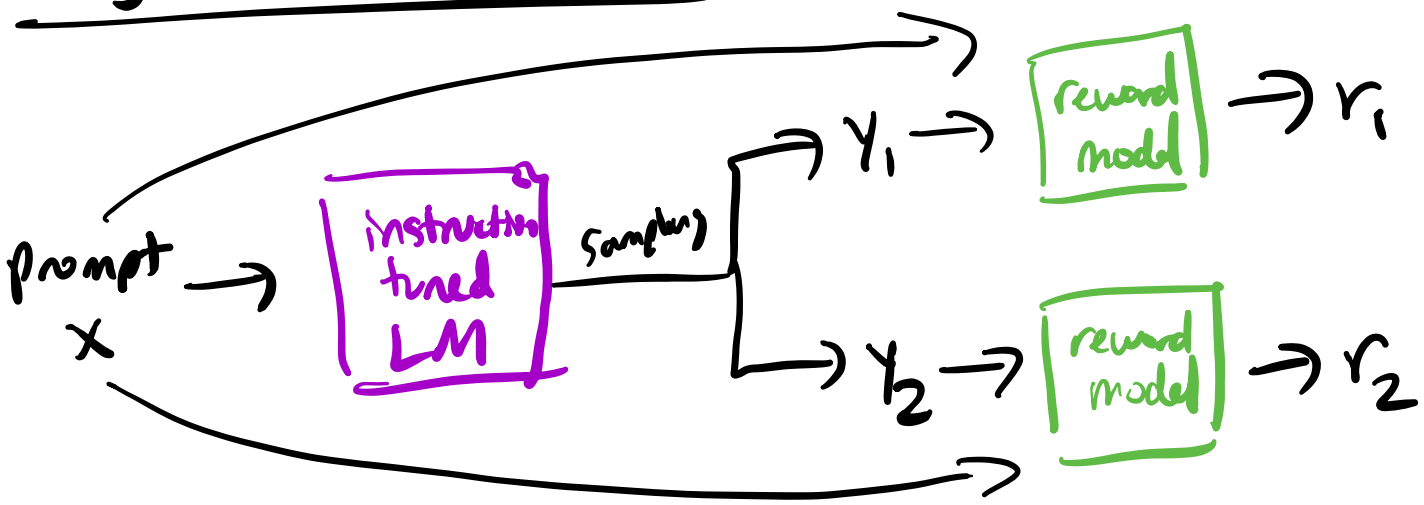


$\rightarrow r = 8.2$

$$r(x, y_w) = f(\underbrace{w}_{1 \times d_{hid}} h_n)$$

→ fine-tune the entire model using the Bradley-Terry loss

Using the reward model:



1. "best of N" sampling (rejection sampling)

   ↳ generate i different samples given x

   choose $\max_i r(x, y_i)$

   ↳ no training

   ↳ very expensive

2. SFT (on policy model) on high reward samples

   maximize $P_\theta(y_w|x)$

   ↳ RAFT, paper

3. use reinforcement learning to increase $P_\theta(y_w|x)$ by a small amount, and decrease $P_\theta(y_l|x)$ ↗ where amounts are functions of $r(x, y)$

→ maximize expected reward of y given x

↳ expectation ⇒ probability-weighted average of rewards

let's say we have prompt x

for all possible outputs y,

$$E[r(x,y)] = \sum_y p_\theta(y|x)\, r(x,y)$$

↳ impossible to enumerate all y

↳ estimate via sampling

↳ use just one y

↳ small number of y's

$$J(\theta) = \mathop{E}_{y \sim p_\theta(\cdot|x)}[r(x,y)]$$

$$= \sum_y p_\theta(y|x)\, r(x,y)$$

$$\frac{dJ}{d\theta} = \sum_y r(x,y) \frac{d}{d\theta} p_\theta(y|x)$$