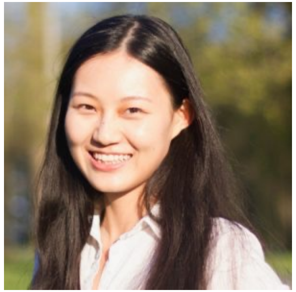
Iterative reasoning preference optimization

Richard Yuanzhe Pang   Weizhe Yuan   Kyunghyun Cho   He He

Sainbayar Sukhbaatar*   Jason Weston*

Meta   NYU ML^2



Iterative preference optimization on general instruction following tasks:

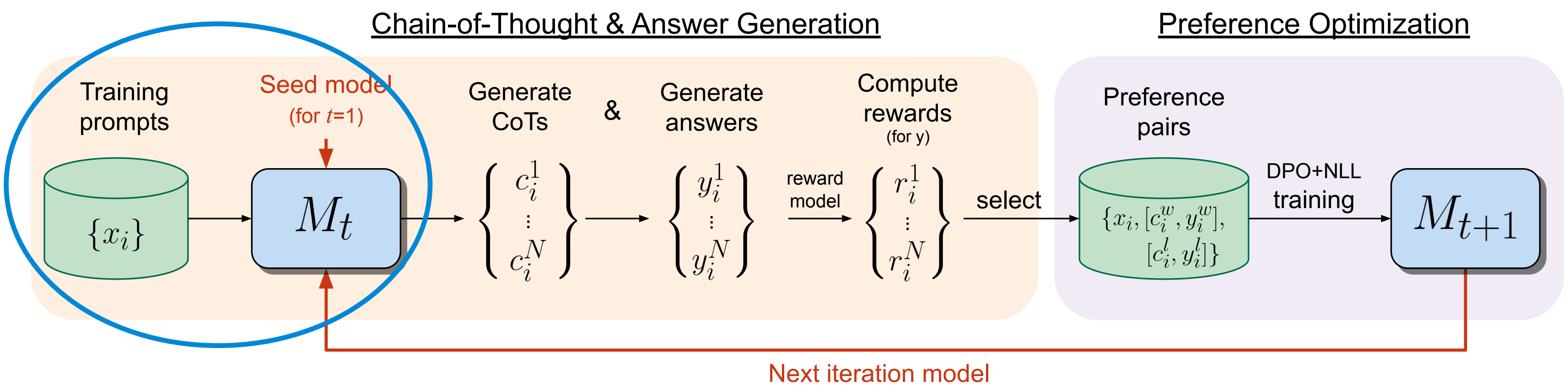
- DPO (Rafailov et al., 2023) → Iterative DPO (Xu et al., 2023)
- Self-rewarding LM (Yuan et al., 2023)
- SPIN (Chen et al., 2024)

Training methods on reasoning:

- STaR (Zelikman et al., 2022)
- ReST^{EM} (Singh et al., 2024)
- V-STaR (Hosseini et al., 2024)

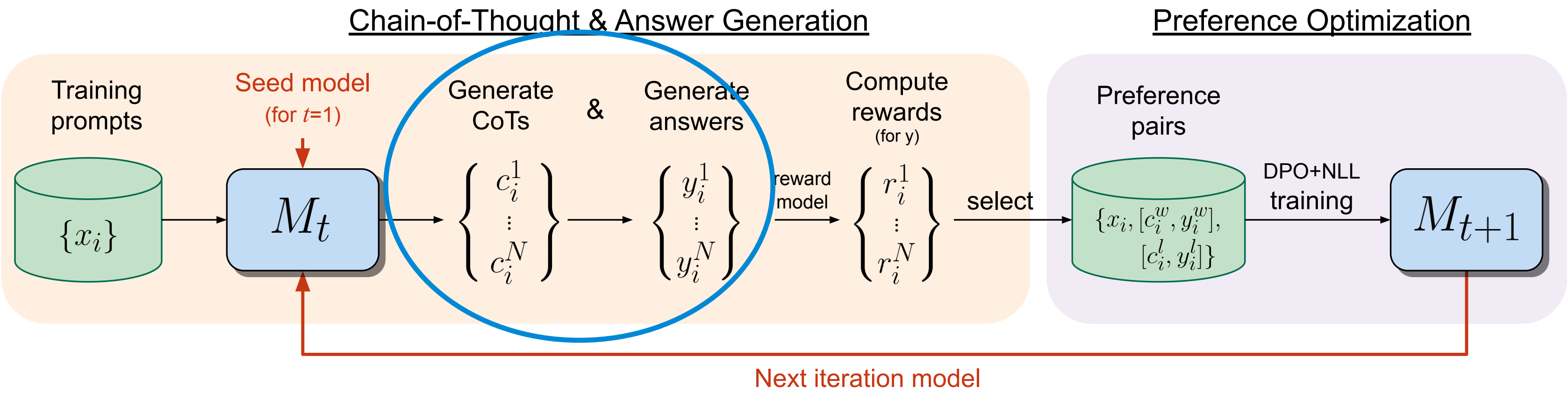
We develop an approach to apply iterative preference optimization to reasoning tasks.

Iterative reasoning preference optimization (IRPO)



Start with base model & fixed training set with labels

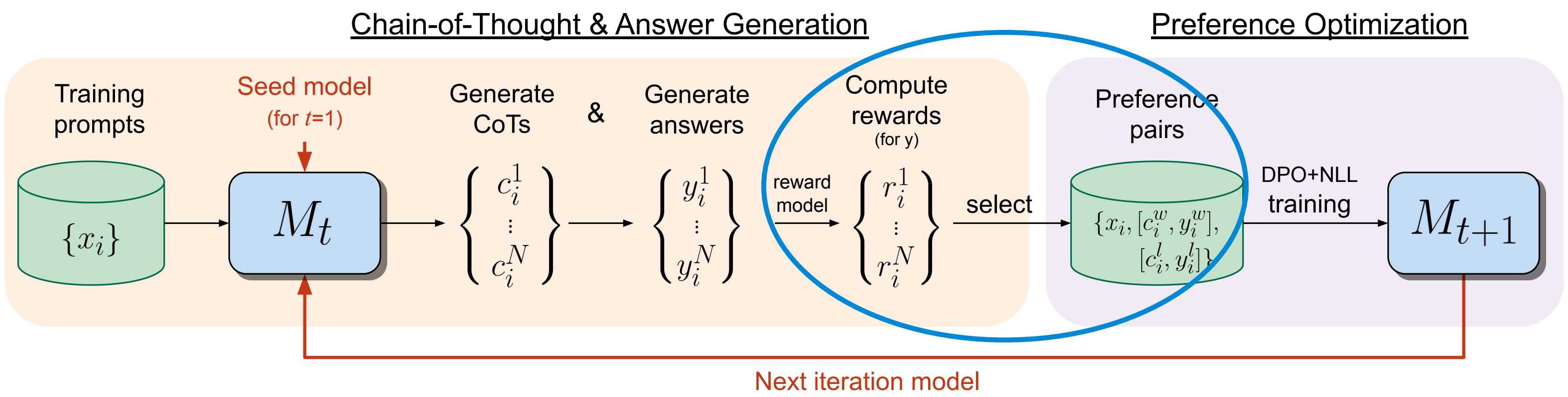
Iterative reasoning preference optimization (IRPO)



Start with base model & fixed training set with labels

- Generate multiple CoTs+answers per train example with current model

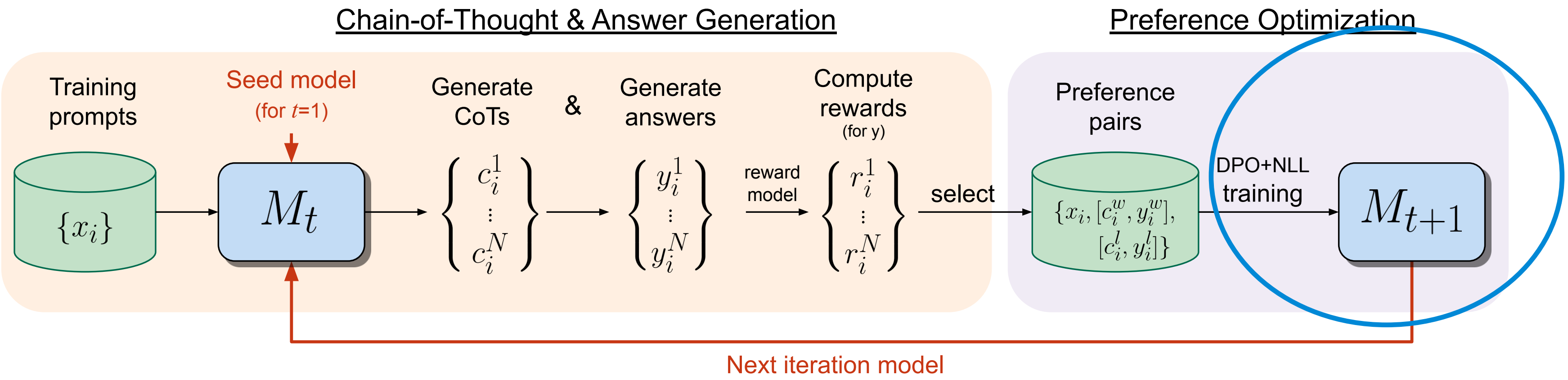
Iterative reasoning preference optimization (IRPO)



Start with base model & fixed training set with labels

- Generate multiple CoTs+answers per train example with current model
- Build preference pairs based on answer correct vs. not

Iterative reasoning preference optimization (IRPO)

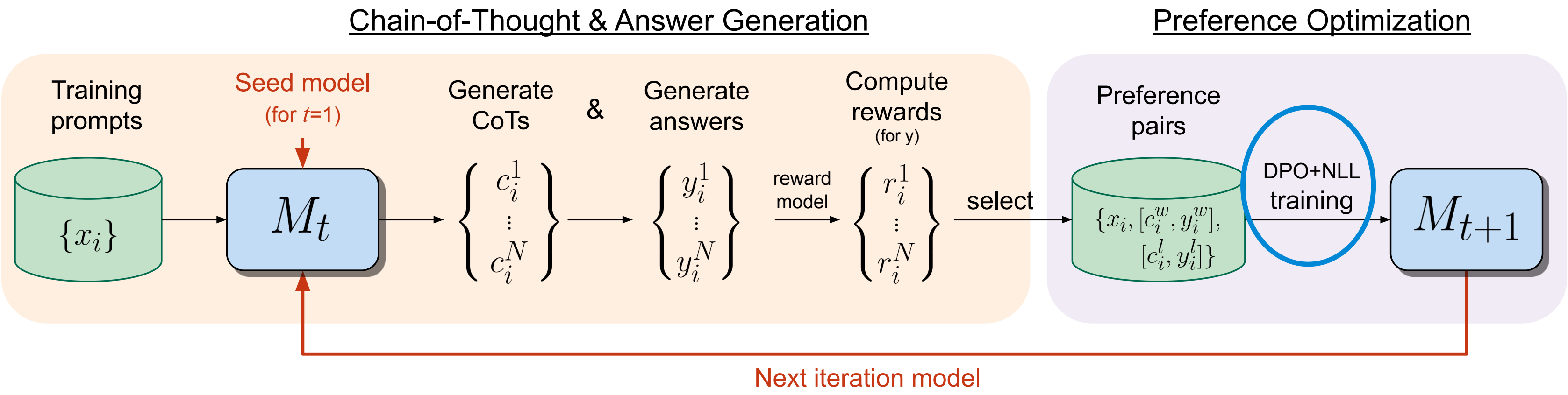


Start with base model & fixed training set with labels

- Generate multiple CoTs+answers per train example with current model
- Build preference pairs based on answer correct vs. not
- Train w/ DPO + NLL term for correct CoTs+answers

Repeat steps with new model

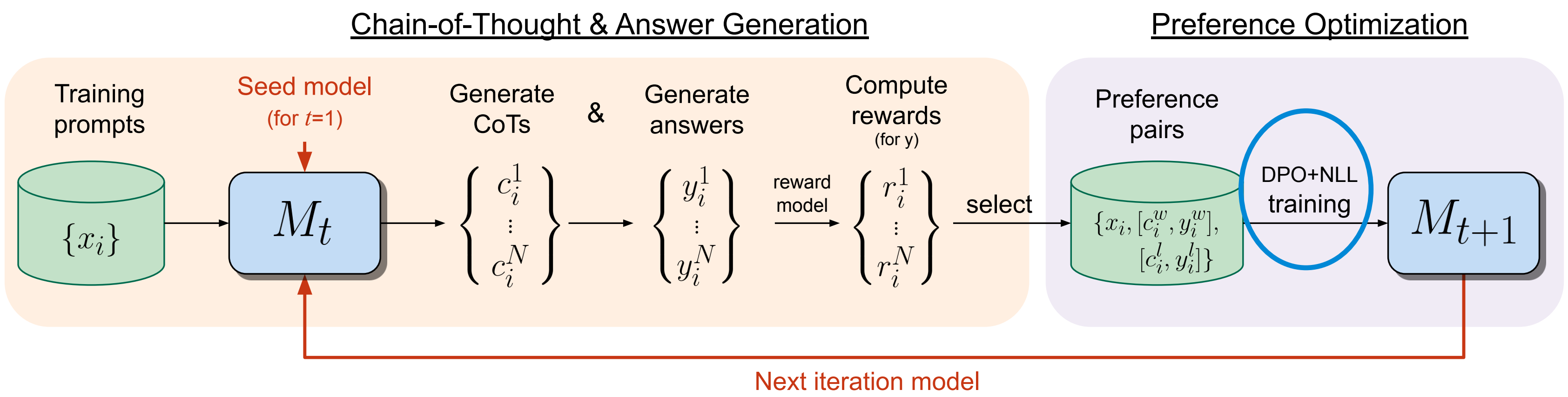
DPO + NLL



$$\mathcal{L}_{\text{DPO}}(c_i^w, y_i^w, c_i^l, y_i^l | x_i)$$

$$= -\log \sigma \left(\beta \log \frac{M_\theta(c_i^w, y_i^w | x_i)}{M_t(c_i^w, y_i^w | x_i)} - \beta \log \frac{M_\theta(c_i^l, y_i^l | x_i)}{M_t(c_i^l, y_i^l | x_i)} \right)$$

DPO + NLL



$$\mathcal{L}_{\text{DPO+NLL}} = \mathcal{L}_{\text{DPO}}(c_i^w, y_i^w, c_i^l, y_i^l | x_i) + \alpha \mathcal{L}_{\text{NLL}}(c_i^w, y_i^w | x_i)$$

$$= -\log \sigma \left(\beta \log \frac{M_\theta(c_i^w, y_i^w | x_i)}{M_t(c_i^w, y_i^w | x_i)} - \beta \log \frac{M_\theta(c_i^l, y_i^l | x_i)}{M_t(c_i^l, y_i^l | x_i)} \right) - \alpha \frac{\log M_\theta(c_i^w, y_i^w | x_i)}{|c_i^w| + |y_i^w|}$$

GSM8K

Zero-shot CoT	55.6
+ majority vote (32 samples)	70.7
SFT on gold CoT	63.5
SFT on generated chosen CoTs (STaR 1 iteration)	65.2
DPO init from llama	61.8
DPO init from SFT model trained on chosen CoTs	60.3

Iterative RPO	
Iteration 1	73.1
Iteration 2	78.0
Iteration 3	81.1
Iteration 4	81.6
+ majority vote (32 samples)	88.7

Init from Llama-2-70b-chat

GSM8K

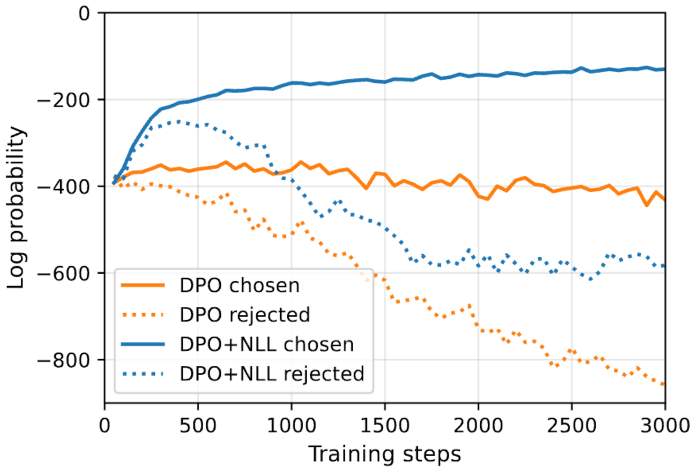
Zero-shot CoT	55.6
+ majority vote (32 samples)	70.7
SFT on gold CoT	63.5
SFT on generated chosen CoTs (STaR 1 iteration)	65.2
DPO init from llama	61.8
DPO init from SFT model trained on chosen CoTs	60.3
SFT on generated chosen CoTs, but on twice as much data	66.9
Iterative RPO (Iteration 1) but on twice as much data	74.8

Iterative RPO	
Iteration 1	73.1
Iteration 2	78.0
Iteration 3	81.1
Iteration 4	81.6
+ majority vote (32 samples)	88.7

ARC-Challenge and MATH

Model	ARC-Challenge (0-shot) Test acc %	MATH (4-shot) Test acc %
<u>Iterative RPO</u>		
Iteration 1	84.8	17.7
Iteration 2	86.2	19.9
Iteration 3	86.7	20.8
+ majority vote (32 samples)	87.9	29.1
<u>Other Llama-2-70b-chat-initialized methods</u>		
CoT	77.8	12.5
SFT *on chosen sequences*	79.8	16.8
DPO *init from Llama-2-70b-chat*	82.8	12.4
DPO *init from SFT model trained on chosen seqs*	83.5	10.5

DPO+NLL



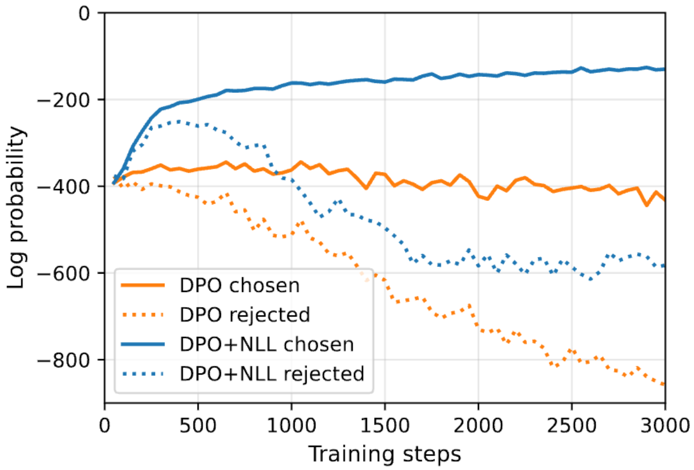
(a) Initialized from Llama

(b) Initialized from SFT trained on chosen seqs

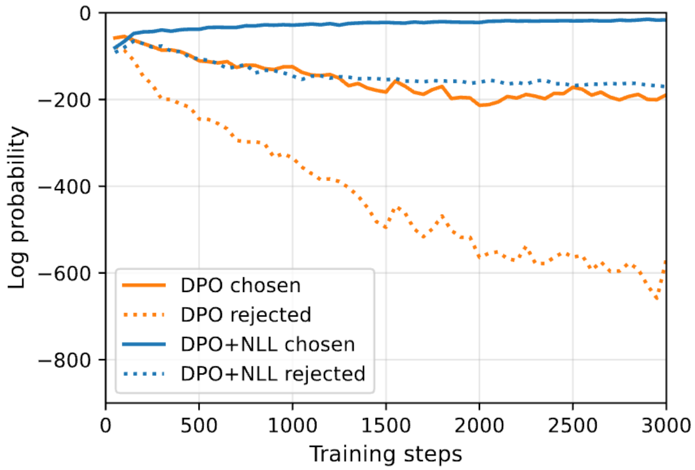
We find the NLL term to be crucial, e.g., GSM8k results 73.1% vs. 61.8%

- Obs 1: margin increasing
- Obs 2: without NLL, both chosen and rejected log probs decrease

DPO+NLL



(a) Initialized from Llama



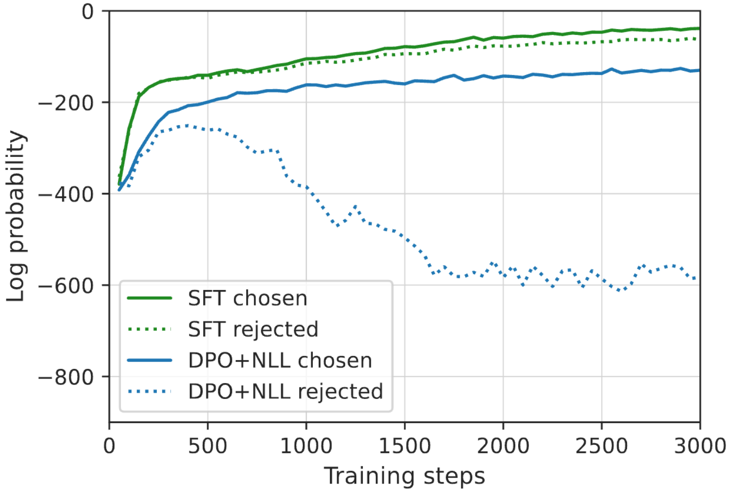
(b) Initialized from SFT trained on chosen seqs

We find the NLL term to be crucial, e.g., GSM8k results 73.1% vs. 61.8%

- Obs 1: margin increasing
- Obs 2: without NLL, both chosen and rejected log probs decrease

Q: What scenario (e.g., sampling approach, task) is naïve DPO harmful on?

Why does SFT not work too well?

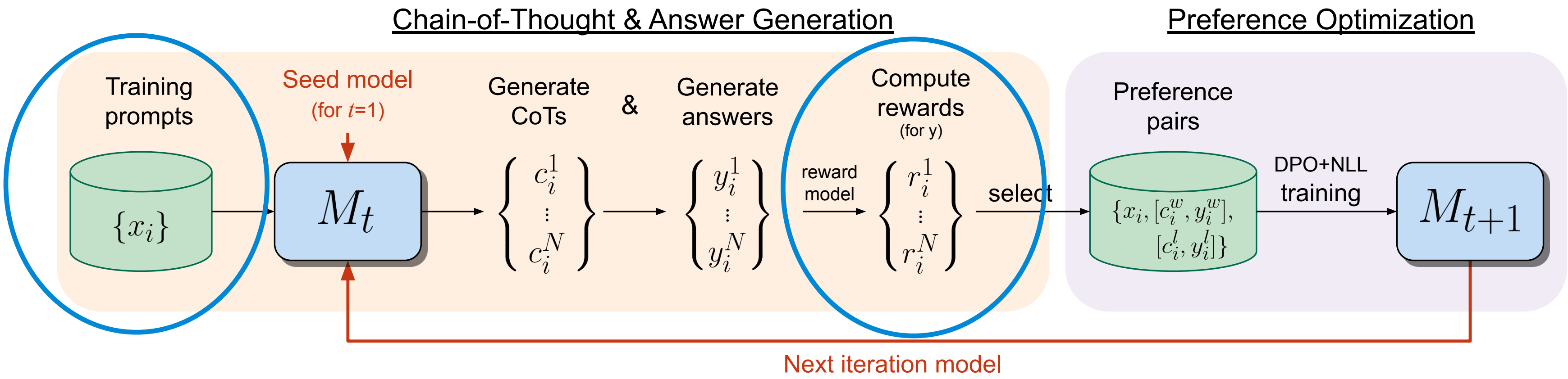


(a) SFT trained on chosen seqs; init from Llama

We find that negative examples are crucial.

- When doing SFT on good sequences, the rejected seqs' probs also go up a lot!

Extension: unsupervised version of IRPO



- What if we generate prompts & don’t know the reference answer?
- Look at consistency – we trust a majority vote answer more if it has a higher proportion of votes
- **Self-Consistency Preference Optimization (ScPO)**

Next steps

- Self-consistency preference optimization (Prasad et al., 2024)
- Figure out when and why naïve DPO does not work
    - Unintentional unalignment (Razin et al., 2024): intuitively, "when $\boldsymbol{y}^{+}$ was No and $\boldsymbol{y}^{-}$ was Never, the probability of Yes would sharply increase
- More iterations for IRPO → on-policy DPO