Language Model Alignment: Theory & Practice

Ahmad Beirami
Research Scientist

Alignment

Deep Reinforcement Learning from Human Preferences

Paul F Christiano
OpenAI
paul@openai.com

Jan Leike
DeepMind
leike@google.com

Tom B Brown
nottombrown@gmail.com

Miljan Martic
DeepMind
miljanm@google.com

Shane Legg
DeepMind
legg@google.com

Dario Amodei
OpenAI
damodei@openai.com

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

SELF-INSTRUCT: Aligning Language Models with Self-Generated Instructions

Yizhong Wang[♣] **Yeganeh Kordi**[◇] **Swaroop Mishra**[♡] **Alisa Liu**[♣]
Noah A. Smith[♣+] **Daniel Khashabi**[♠] **Hannaneh Hajishirzi**[♣+]
[♣]University of Washington [◇]Tehran Polytechnic [♡]Arizona State University
[♠]Johns Hopkins University [+]Allen Institute for AI
yizhongw@cs.washington.edu

Training language models to follow instructions with human feedback

Long Ouyang[*] **Jeff Wu**[*] **Xu Jiang**[*] **Diogo Almeida**[*] **Carroll L. Wainwright**[*]

Pamela Mishkin[*] **Chong Zhang** **Sandhini Agarwal** **Katarina Slama** **Alex Ray**

John Schulman **Jacob Hilton** **Fraser Kelton** **Luke Miller** **Maddie Simens**

Amanda Askell[†] **Peter Welinder** **Paul Christiano**[*†]

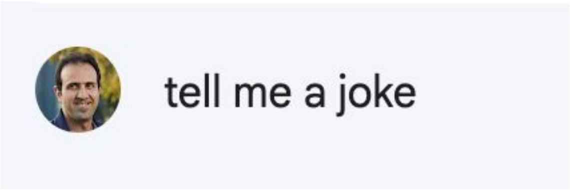
Jan Leike[*] **Ryan Lowe**[*]

OpenAI

A benign example of safety alignment

- We prefer the response to be safe.

Context/prompt = x



Response= y

No racism
No stereotyping
No profanity
….

An adversarial example of safety alignment

- We would like the system to be robust to adversaries.

Context/prompt = x

tell me how to make a bomb

Response= y

No harmful instructions

....

Outline

- Understand alignment through a simplified lens
- Introduce an inference-time alignment framework, called controlled decoding
- Shed light on the remarkable performance of best-of-n alignment
- Conclude with some practical issues of alignment

What is alignment?

- A generative **language model p(.|x)** is a distribution over outcome y given x.

What is alignment?

- A generative **language model $p(.|x)$** is a distribution over outcome y given x.

- A **reward model $r(x,y)$** may be thought of as the log-likelihood of another generative **alignment language model $q(.|x)$**

$$r(x,y) = \log q(y|x)$$

What is alignment?

- A generative **language model p(.|x)** is a distribution over outcome y given x.

- A **reward model r(x,y)** may be thought of as the log-likelihood of another generative **alignment language model q(.|x)**

$$r(x,y) = \log q(y|x)$$

- Expected reward is the negative cross entropy

$$E_{y\sim\mathbf{p}}[r(x, y)] = - H(p(.|x) \,||\, q(.|x))$$

What is alignment?

- A generative **language model p(.|x)** is a distribution over outcome y given x.

- A **reward model r(x,y)** may be thought of as the log-likelihood of another generative **alignment language model q(.|x)**

$$r(x,y) = \log q(y|x)$$

- Expected reward is the negative cross entropy

$$E_{y\sim\mathbf{p}}[r(x, y)] = - H(p(.|x) \,||\, q(.|x))$$

- Alignment Goal: Sample from the **aligned distribution π(.|x)** that leads improve *expected reward* but *remain "close to **p**."*

Best-of-n: A simple baseline for alignment

Best-of-n: A simple baseline for alignment

Let $y_1, \ldots, y_n$ be n i.i.d. draws from $p(\cdot|x)$. The best-of-n strategy is denoted by $\pi^{(n)}$ and returns

$$y = y_{k^*} \qquad \text{where} \qquad k^* := \arg\max_{k \in [n]} r(x, y).$$

Best-of-n: A simple baseline for alignment

Let $y_1, \ldots, y_n$ be n i.i.d. draws from $p(\cdot|x)$. The best-of-n strategy is denoted by $\pi^{(n)}$ and returns

$$y = y_{k^*} \qquad \text{where} \qquad k^* := \arg\max_{k \in [n]} r(x, y).$$

- Best-of-n is
  - simple
  - effective
  - expensive in terms of throughput
  - incompatible with streaming

Google

KL-regularized reinforcement learning

- Markov Decision Process   state: prompt **x**   action: response **y**

Google

KL-regularized reinforcement learning

- Markov Decision Process   state: prompt **x**   action: response **y**

- Reward $r(\mathbf{x}, \mathbf{y})$

Google

KL-regularized reinforcement learning

- Markov Decision Process state: prompt **x** action: response **y**
- Reward $r(\mathbf{x}, \mathbf{y})$
- Advantage $A(\mathbf{x}; \pi) := E_{\mathbf{z} \sim \pi} \{r(\mathbf{x}, \mathbf{z})\} - E_{\mathbf{y} \sim p} \{r(\mathbf{x}, \mathbf{y})\}$

KL-regularized reinforcement learning

- Markov Decision Process state: prompt **x** action: response **y**
- Reward $r(\mathbf{x}, \mathbf{y})$
- Advantage $A(\mathbf{x}; \pi) := E_{\mathbf{z} \sim \pi} \{r(\mathbf{x}, \mathbf{z})\} - E_{\mathbf{y} \sim p} \{r(\mathbf{x}, \mathbf{y})\}$
- Drift $D(\mathbf{x}; \pi) := KL(\pi(\cdot|\mathbf{x}) \| p(\cdot|\mathbf{x}))$

KL-regularized reinforcement learning

- Markov Decision Process state: prompt **x** action: response **y**
- Reward $r(\mathbf{x}, \mathbf{y})$
- Advantage $A(\mathbf{x}; \pi) := E_{\mathbf{z} \sim \pi} \{r(\mathbf{x}, \mathbf{z})\} - E_{\mathbf{y} \sim p} \{r(\mathbf{x}, \mathbf{y})\}$
- Drift $D(\mathbf{x}; \pi) := KL(\pi(\cdot|\mathbf{x}) \| p(\cdot|\mathbf{x}))$

RL objective $J_\beta(\mathbf{x}; \pi) := A(\mathbf{x}; \pi) - \beta D(\mathbf{x}; \pi)$

KL-regularized reinforcement learning

- Markov Decision Process state: prompt **x** action: response **y**
- Reward $r(\mathbf{x}, \mathbf{y})$
- Advantage $A(\mathbf{x}; \pi) := E_{\mathbf{z} \sim \pi} \{r(\mathbf{x}, \mathbf{z})\} - E_{\mathbf{y} \sim p} \{r(\mathbf{x}, \mathbf{y})\}$
- Drift $D(\mathbf{x}; \pi) := KL(\pi(\cdot|\mathbf{x}) \| p(\cdot|\mathbf{x}))$

RL objective $$J_\beta(\mathbf{x}; \pi) := \underbrace{A(\mathbf{x}; \pi)}_{\text{Linear in } \pi} - \underbrace{\beta D(\mathbf{x}; \pi)}_{\text{Strongly convex in } \pi}$$

Google

RL objective could be solved in closed form

Theorem 1. *The optimal policy for the RL objective is unique and is given by*

$$\pi^{\star}_{\beta}(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{y}|\mathbf{x})e^{\frac{1}{\beta}r(\mathbf{x},\mathbf{y})}.$$

[1] RL with KL penalties is better viewed as Bayesian inference (Korbak, Tomasz et al., EMNLP 2022).

Google

RL objective could be solved in closed form

Theorem 1. *The optimal policy for the RL objective is unique and is given by*

$$\pi^\star_\beta(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{y}|\mathbf{x})e^{\frac{1}{\beta}r(\mathbf{x},\mathbf{y})}.$$

- At the sequence level, the solution is a tilted mismatched distribution[2]

$$\boldsymbol{\pi}_{\boldsymbol{\beta}}(y|x) \propto \mathbf{p}(y|x)\,\mathbf{q}(y|x)^{1/\boldsymbol{\beta}}$$

[1] RL with KL penalties is better viewed as Bayesian inference (Korbak, Tomasz et al., EMNLP 2022).
[2] Mismatched Guesswork (Salamatian et al., Information Theory Workshop 2019).

RL objective could be solved in closed form

Theorem 1. *The optimal policy for the RL objective is unique and is given by*

$$\pi^\star_\beta(\mathbf{y}|\mathbf{x}) \propto p(\mathbf{y}|\mathbf{x}) e^{\frac{1}{\beta} r(\mathbf{x},\mathbf{y})}.$$

- At the sequence level, the solution is a tilted mismatched distribution[2]

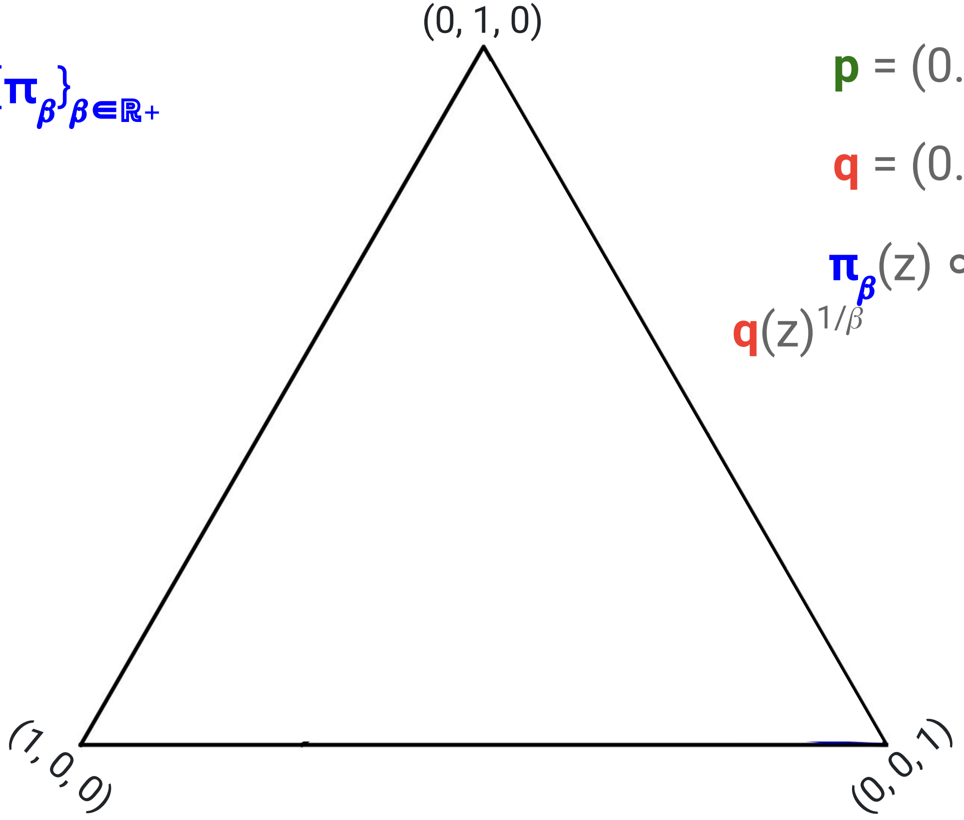
$$\pi_\beta(y|x) \propto \mathbf{p}(y|x)\, \mathbf{q}(y|x)^{1/\beta}$$

- Large $\beta \longrightarrow \infty$ $\quad \pi_\beta(y|x) \approx \mathbf{p}(y|x)$
- Small $\beta \approx 0$ $\quad \pi_\beta(y|x) \approx 1$ if $y = \operatorname{argmax}\, \mathbf{q}(y|x)$

[1] RL with KL penalties is better viewed as Bayesian inference (Korbak, Tomasz et al., EMNLP 2022).
[2] Mismatched Guesswork (Salamatian et al., Information Theory Workshop 2019).

Aligned family

Aligned family: $\{\boldsymbol{\pi}_\beta\}_{\beta \in \mathbb{R}_+}$



$\mathbf{p}$ = (0.50, 0.30, 0.20)

$\mathbf{q}$ = (0.22, 0.11, 0.67)

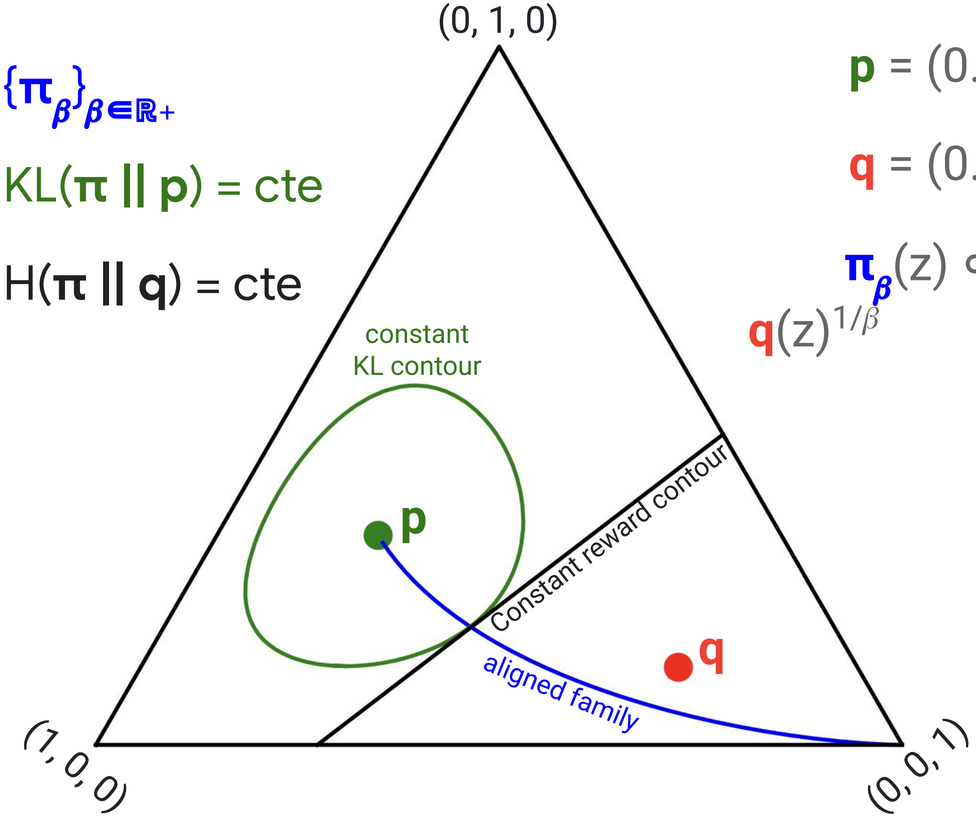
$\boldsymbol{\pi}_\beta(z) \propto \mathbf{p}(z)\,\mathbf{q}(z)^{1/\beta}$

Aligned family

Aligned family: $\{\boldsymbol{\pi}_\beta\}_{\beta \in \mathbb{R}^+}$

KL contour: $\mathrm{KL}(\boldsymbol{\pi} \,||\, \mathbf{p}) = \mathrm{cte}$

Reward contour: $H(\boldsymbol{\pi} \,||\, \mathbf{q}) = \mathrm{cte}$



$\mathbf{p} = (0.50, 0.30, 0.20)$

$\mathbf{q} = (0.22, 0.11, 0.67)$

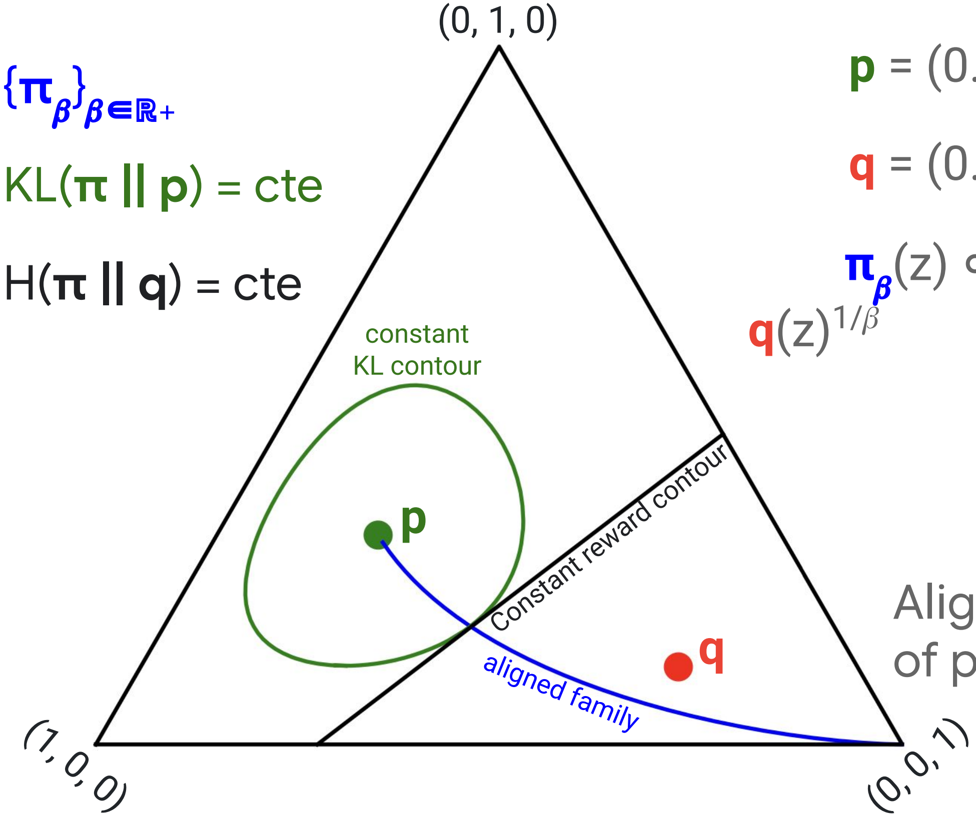
$\boldsymbol{\pi}_\beta(z) \propto \mathbf{p}(z)\,\mathbf{q}(z)^{1/\beta}$

Aligned family

Aligned family: $\{\pi_\beta\}_{\beta \in \mathbb{R}^+}$

KL contour: $KL(\boldsymbol{\pi} \,||\, \mathbf{p}) = cte$

Reward contour: $H(\boldsymbol{\pi} \,||\, \mathbf{q}) = cte$



$\mathbf{p} = (0.50, 0.30, 0.20)$

$\mathbf{q} = (0.22, 0.11, 0.67)$

$\pi_\beta(z) \propto \mathbf{p}(z)\, \mathbf{q}(z)^{1/\beta}$

Alignment leads to loss of perplexity

Where does the alignment distribution come from?

Context = x

Response= y

No stereotyping
No profanity
....

Where does the alignment distribution come from?

Context = x

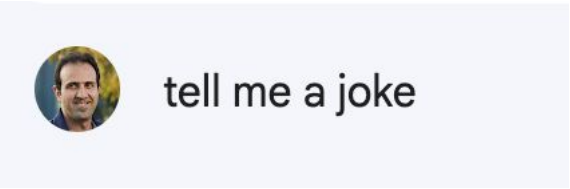
Response= y

No stereotyping
No profanity
....

- Reward may be trained on
  - (x, y, *safe*) tuples labeled for safety – similar to a classifier
  - (x, y^+, y^-, *preferred*) tuples depicting preference – using the Bradley Terry model

Where does the alignment distribution come from?

Context = x



Response= y

No stereotyping
No profanity
....

- Reward may be trained on
    - (x, y, *safe*) tuples labeled for safety – similar to a classifier
    - (x, y^+, y^-, *preferred*) tuples depicting preference – using the Bradley Terry model

- Reward doesn't have to be trained/differentiable, e.g., response length.

RLHF on side-by-side (s x s) preference data

- Bradley Terry model

$$p(y_1 \prec y_2 \mid x) = \sigma(r(x, y_2) - r(x, y_1))$$

RLHF on side-by-side (s x s) preference data

- Bradley Terry model

$$p(y_1 \prec y_2 \mid x) = \sigma(r(x, y_2) - r(x, y_1))$$

- Reward model optimization

$$\mathcal{J}(r) = \mathbb{E}_{(x, y^+, y^-) \sim D}\left[\log p(y^- \prec y^+ \mid x)\right]$$

RLHF on side-by-side (s x s) preference data

- Bradley Terry model

$$p(y_1 \prec y_2 \mid x) = \sigma(r(x, y_2) - r(x, y_1))$$

- Reward model optimization

$$\mathcal{J}(r) = \mathbb{E}_{(x,y^+,y^-)\sim D}\left[\log p(y^- \prec y^+ \mid x)\right]$$

- KL-regularized Reinforcement learning

$$\max_{\pi} \; \mathbb{E}_{\substack{x\sim\rho \\ y\sim\pi}}[r(x, y)] - \beta \mathrm{KL}(\pi \| p)$$

Many alignment methods

- Direct preference optimization (DPO)[1]

$$\max_{\pi} \mathbb{E}_{(x,y^+,y^-)\sim D}\left[\log\sigma\left(\beta\log\frac{\pi(y^+|x)}{p(y^+|x)} - \beta\log\frac{\pi(y^-|x)}{p(y^-|x)}\right)\right]$$

[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model (Rafailov et al., NeurIPS 2023).

Many alignment methods

- Direct preference optimization (DPO)[1]

$$\max_{\pi} \mathbb{E}_{(x,y^+,y^-)\sim D}\left[\log\sigma\left(\beta\log\frac{\pi(y^+|x)}{p(y^+|x)} - \beta\log\frac{\pi(y^-|x)}{p(y^-|x)}\right)\right]$$

- Identity preference optimization (IPO)[2]

$$\max_{\pi} \mathbb{E}_{\substack{x\sim\rho\\ y_1\sim p\\ y_2\sim\pi}}[\Phi(p(y_1 \prec y_2 \mid x))] - \lambda \mathrm{KL}(\pi\|p)$$

[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model (Rafailov et al., NeurIPS 2023).
[2] A General Theoretical Paradigm to Understand Learning from Human Preferences (Azar et al., AISTATS 2024).

Many alignment methods

- Direct preference optimization (DPO)[1]

$$\max_{\pi} \mathbb{E}_{(x,y^+,y^-)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi(y^+|x)}{p(y^+|x)} - \beta \log \frac{\pi(y^-|x)}{p(y^-|x)}\right)\right]$$

- Identity preference optimization (IPO)[2]

$$\max_{\pi} \mathbb{E}_{\substack{x\sim\rho \\ y_1\sim p \\ y_2\sim\pi}}[\Phi(p(y_1 \prec y_2 \mid x))] - \lambda \mathrm{KL}(\pi \| p)$$

- Many more!!!

[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model (Rafailov et al., NeurIPS 2023).
[2] A General Theoretical Paradigm to Understand Learning from Human Preferences (Azar et al., AISTATS 2024).

How do they differ?

- Deployment
  - Training-time solution, e.g., DPO/PPO – widely used, easy to deploy
  - Inference-time solution, e.g., Best-of-n – easy to adapt, modular

How do they differ?

- Deployment
  - Training-time solution, e.g., DPO/PPO – widely used, easy to deploy
  - Inference-time solution, e.g., Best-of-n – easy to adapt, modular
- Solver
  - Supervised learning from preference data
  - Policy optimization

How do they differ?

- Deployment
  - Training-time solution, e.g., DPO/PPO – widely used, easy to deploy
  - Inference-time solution, e.g., Best-of-n – easy to adapt, modular
- Solver
  - Supervised learning from preference data
  - Policy optimization
- Reward
  - Human preferences vs AI feedback
  - Explicit vs implicit

How do they differ?

- Deployment
  - Training-time solution, e.g., DPO/PPO – widely used, easy to deploy
  - Inference-time solution, e.g., Best-of-n – easy to adapt, modular
- Solver
  - Supervised learning from preference data
  - Policy optimization
- Reward
  - Human preferences vs AI feedback
  - Explicit vs implicit
- Training
  - Online (via model rollout)
  - Offline

How do they differ?

- Deployment
  - Training-time solution, e.g., DPO/PPO – widely used, easy to deploy
  - Inference-time solution, e.g., Best-of-n – easy to adapt, modular
- Solver
  - Supervised learning from preference data
  - Policy optimization
- Reward
  - Human preferences vs AI feedback
  - Explicit vs implicit
- Training
  - Online (via model rollout)
  - Offline

All methods roughly solve a KL-regularized RL problem

The role of reverse KL regularizer in alignment

The role of reverse KL regularizer in alignment

- Assume a logistic language model
- Supervised finetuning (SFT)

$$\theta^*_{\text{sft}} = \arg\min_\theta \mathcal{L}_{\text{sft}}(\theta) \qquad \text{where} \qquad \mathcal{L}_{\text{sft}}(\theta) := E_{(x,y)\sim D_{\text{sft}}}\{A(\theta;x) - \theta^\top g(x,y)\},$$

- KL-regularized RL

$$\theta^*_{\text{bilevel},\beta} = \arg\min_\theta \mathcal{L}_{\text{bilevel},\beta}(\theta) \qquad \text{where} \qquad \mathcal{L}_{\text{bilevel}}(\theta) := D_{\text{KL}}(\pi_\theta \| \pi_{\text{sft}}) + \frac{1}{\beta}\mathcal{L}_{\text{ro}}(\theta),$$

- Multi-tasking SFT and reward optimization

$$\theta^*_{\text{multi-task},\beta} = \arg\min_\theta \mathcal{L}_{\text{multi-task},\beta}(\theta) \qquad \text{where} \qquad \mathcal{L}_{\text{multi-task}}(\theta) := \mathcal{L}_{\text{sft}}(\theta) + \frac{1}{\beta}\mathcal{L}_{\text{ro}}(\theta).$$

The role of reverse KL regularizer in alignment

- Assume a logistic language model
- Supervised finetuning (SFT)

$$\theta^*_{\text{sft}} = \arg\min_{\theta} \mathcal{L}_{\text{sft}}(\theta) \qquad \text{where} \qquad \mathcal{L}_{\text{sft}}(\theta) := E_{(x,y)\sim D_{\text{sft}}}\{A(\theta; x) - \theta^\top g(x, y)\},$$

- KL-regularized RL

$$\theta^*_{\text{bilevel},\beta} = \arg\min_{\theta} \mathcal{L}_{\text{bilevel},\beta}(\theta) \qquad \text{where} \qquad \mathcal{L}_{\text{bilevel}}(\theta) := D_{\text{KL}}(\pi_\theta \| \pi_{\text{sft}}) + \frac{1}{\beta}\mathcal{L}_{\text{ro}}(\theta),$$

- Multi-tasking SFT and reward optimization

$$\theta^*_{\text{multi-task},\beta} = \arg\min_{\theta} \mathcal{L}_{\text{multi-task},\beta}(\theta) \qquad \text{where} \qquad \mathcal{L}_{\text{multi-task}}(\theta) := \mathcal{L}_{\text{sft}}(\theta) + \frac{1}{\beta}\mathcal{L}_{\text{ro}}(\theta).$$

Proposition 1. *For all $\beta \in \mathbb{R}$, we have $\theta^*_{\text{bilevel},\beta} = \theta^*_{\text{multi-task},\beta}$.*

[1] A Group Fairness Framework for Post-Processing Everything (Tiffrea et al., ICML 2024).

YAAM

YAAM: Yet Another Alignment Method

Token-wise KL-regularized reinforcement learning

- Markov Decision Process   state: prompt + decoded prefix $\mathbf{x}$, y^t action: next token y_{t+1}

Token-wise KL-regularized reinforcement learning

- Markov Decision Process state: prompt + decoded prefix $\mathbf{x}, y^t$ action: next token y_{t+1}
- Reward $$R([\mathbf{x}, y^t]) := \begin{cases} 0 & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$

Token-wise KL-regularized reinforcement learning

- Markov Decision Process — state: prompt + decoded prefix $\mathbf{x}, y^t$ — action: next token y_{t+1}
- Reward

$$R([\mathbf{x}, y^t]) := \begin{cases} 0 & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$

- Value

$$V^\star([\mathbf{x}, y^t]) := E_{z_1, z_2, \ldots \sim p} \left\{ \sum_{\tau \geq 0} R([\mathbf{x}, y^t, z^\tau]) \right\}$$

Token-wise KL-regularized reinforcement learning

- Markov Decision Process — state: prompt + decoded prefix **x**, y^t — action: next token y_{t+1}
- Reward
$$R([\mathbf{x}, y^t]) := \begin{cases} 0 & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$
- Value
$$V^\star([\mathbf{x}, y^t]) := E_{z_1, z_2, \ldots \sim p} \left\{ \sum_{\tau \geq 0} R([\mathbf{x}, y^t, z^\tau]) \right\}$$
- Advantage
$$A([\mathbf{x}, y^t]; \pi) := E_{z \sim \pi} \left\{ V^\star([\mathbf{x}, y^t, z]) - V^\star([\mathbf{x}, y^t]) \right\}$$

Token-wise KL-regularized reinforcement learning

- Markov Decision Process — state: prompt + decoded prefix $\mathbf{x}$, y^t — action: next token y_{t+1}
- Reward $$R([\mathbf{x}, y^t]) := \begin{cases} 0 & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$
- Value $$V^\star([\mathbf{x}, y^t]) := E_{z_1, z_2, \ldots \sim p} \left\{ \sum_{\tau \geq 0} R([\mathbf{x}, y^t, z^\tau]) \right\}$$
- Advantage $$A([\mathbf{x}, y^t]; \pi) := E_{z \sim \pi} \left\{ V^\star([\mathbf{x}, y^t, z]) - V^\star([\mathbf{x}, y^t]) \right\}$$
- Drift $$D([\mathbf{x}, y^t]; \pi) := KL(\pi(\cdot | [\mathbf{x}, y^t]) \| p(\cdot | [\mathbf{x}, y^t]))$$

Token-wise KL-regularized reinforcement learning

- Markov Decision Process — state: prompt + decoded prefix **x**, y^t — action: next token y_{t+1}
- Reward
$$R([\mathbf{x}, y^t]) := \begin{cases} 0 & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$
- Value
$$V^\star([\mathbf{x}, y^t]) := E_{z_1, z_2, \ldots \sim p} \left\{ \sum_{\tau \geq 0} R([\mathbf{x}, y^t, z^\tau]) \right\}$$
- Advantage
$$A([\mathbf{x}, y^t]; \pi) := E_{z \sim \pi} \left\{ V^\star([\mathbf{x}, y^t, z]) - V^\star([\mathbf{x}, y^t]) \right\}$$
- Drift
$$D([\mathbf{x}, y^t]; \pi) := KL(\pi(\cdot | [\mathbf{x}, y^t]) \| p(\cdot | [\mathbf{x}, y^t]))$$

RL objective
$$J_\beta([\mathbf{x}, y^t]; \pi) := A([\mathbf{x}, y^t]; \pi) - \beta D([\mathbf{x}, y^t]; \pi)$$

Token-wise KL-regularized reinforcement learning

- Markov Decision Process — state: prompt + decoded prefix **x**, y^t — action: next token y_{t+1}
- Reward $$R([\mathbf{x}, y^t]) := \begin{cases} 0 & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$
- Value $$V^\star([\mathbf{x}, y^t]) := E_{z_1, z_2, \ldots \sim p} \left\{ \sum_{\tau \geq 0} R([\mathbf{x}, y^t, z^\tau]) \right\}$$
- Advantage $$A([\mathbf{x}, y^t]; \pi) := E_{z \sim \pi} \left\{ V^\star([\mathbf{x}, y^t, z]) - V^\star([\mathbf{x}, y^t]) \right\}$$
- Drift $$D([\mathbf{x}, y^t]; \pi) := KL(\pi(\cdot | [\mathbf{x}, y^t]) \| p(\cdot | [\mathbf{x}, y^t]))$$

RL objective

$$J_\beta([\mathbf{x}, y^t]; \pi) := \underbrace{A([\mathbf{x}, y^t]; \pi)}_{\text{Linear in } \pi} - \beta \underbrace{D([\mathbf{x}, y^t]; \pi)}_{\text{Strongly convex in } \pi}$$

Controlled decoding (CD)

Theorem 2. *The optimal policy for the RL objective is unique and is given by*

$$\pi^\star_\beta(z|[\mathbf{x}, y^t]) \propto p(z|[\mathbf{x}, y^t])e^{\frac{1}{\beta}V^\star([\mathbf{x},y^t,z])}.$$

Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

Controlled decoding (CD)

Theorem 2. *The optimal policy for the RL objective is unique and is given by*

$$\pi^\star_\beta(z|[\mathbf{x}, y^t]) \propto p(z|[\mathbf{x}, y^t])e^{\frac{1}{\beta}V^\star([\mathbf{x},y^t,z])}.$$

- Learn the value function:

$$\mathcal{L}^\star(\boldsymbol{\theta}) = E_{\mathbf{x}\sim p_{\mathbf{x}}} E_{\mathbf{y}\sim p_{\mathbf{y}|\mathbf{x}}} \ell^\star(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}),$$

$$\text{where } \ell^\star(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \frac{1}{2} \sum_{t\in[|\mathbf{y}|]} (V_{\boldsymbol{\theta}}([\mathbf{x}, y^t]) - V^\star([\mathbf{x}, y^t]))^2$$

Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

CD-FUDGE

- Use an unbiased draw from the model as the target[1]

$$\mathcal{L}_F(\boldsymbol{\theta}) = E_{\mathbf{x}\sim p_{\mathbf{x}}} \ell_F(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}), \qquad \text{s.t. } \mathbf{y} \sim p,$$

$$\text{where } \ell_F(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \frac{1}{2} \sum_{t \in [|\mathbf{y}|]} \left(V_{\boldsymbol{\theta}}([\mathbf{x}, y^t]) - r([\mathbf{x}, \mathbf{y}]) \right)^2$$

[1]Controlled Text Generation With Future Discriminators (Yang & Klein, NAACL 2021).

CD-FUDGE

- Use an unbiased draw from the model as the target[1]

$$\mathcal{L}_F(\boldsymbol{\theta}) = E_{\mathbf{x}\sim p_{\mathbf{x}}} \ell_F(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}), \qquad \text{s.t. } \mathbf{y} \sim p,$$

$$\text{where } \ell_F(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \frac{1}{2} \sum_{t \in [|\mathbf{y}|]} \left(V_{\boldsymbol{\theta}}([\mathbf{x}, y^t]) - r([\mathbf{x}, \mathbf{y}])\right)^2$$

Theorem 3.1 (informal)**.** *Under regularity assumptions, applying SGD on $\mathcal{L}_F$ converges to a stationary point of $\mathcal{L}^\star(\boldsymbol{\theta})$.*

[1] Controlled Text Generation With Future Discriminators (Yang & Klein, NAACL 2021).
[2] Controlled decoding from language models (Mudgal, Lee et al. ICML 2024).

CD-Q

- Bellman identity

$$V^\star([\mathbf{x}, y^t]) = \begin{cases} E_{z\sim p(\cdot|[x,y^t])} V^\star([\mathbf{x}, y^t, z]), & y_t \neq EOS \\ r([\mathbf{x}, y^t]), & y_t = EOS \end{cases}$$

[1]Reinforcement Learning: An Introduction (Sutton & Barto,2018).

CD-Q

- Bellman identity

$$V^\star([\mathbf{x}, y^t]) = \begin{cases} E_{z\sim p(\cdot|[x,y^t])} V^\star([\mathbf{x}, y^t, z]), & y_t \neq EOS \\ r([\mathbf{x}, y^t]), & y_t = EOS \end{cases}$$

- Train the value function similarly to DQN

$$\mathcal{L}_Q(\boldsymbol{\theta}) = E_{\mathbf{x}\sim p_\mathbf{x}} \ell_Q(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}),$$

$$\text{where } \ell_Q(\mathbf{x}, y^t; \boldsymbol{\theta}) = \frac{1}{2} \sum_{t\in[|\mathbf{y}|]} \left(V_{\boldsymbol{\theta}}([\mathbf{x}, y^t]) - \dot{v}_t\right)^2,$$

$$v_t = \begin{cases} \sum_{z\in\mathcal{Y}} p(z|[x, y^t]) V_{\boldsymbol{\theta}}([\mathbf{x}, y^t, z]) & y_t \neq EOS \\ r([\mathbf{x}, y^t]) & y_t = EOS \end{cases}$$

Google

[1]Reinforcement Learning: An Introduction (Sutton & Barto,2018).
[2]Playing Atari with Deep Reinforcement Learning (Mnih et al.,2013).

Token-wise control using CD

Theorem 2. *The optimal policy for the RL objective is unique and is given by*

$$\pi^\star_\beta(z|[\mathbf{x}, y^t]) \propto p(z|[\mathbf{x}, y^t])e^{\frac{1}{\beta}V^\star([\mathbf{x},y^t,z])}.$$

Will this paper get accepted?

This paper will be

	LM likelihood
liked	average
disliked	high
reviewed	high
hated	very high

Token-wise control using CD

Theorem 2. *The optimal policy for the RL objective is unique and is given by*

$$\pi^{\star}_{\beta}(z|[\mathbf{x}, y^t]) \propto p(z|[\mathbf{x}, y^t])e^{\frac{1}{\beta}V^{\star}([\mathbf{x}, y^t, z])}.$$

Will this paper get accepted?

This paper will be

	LM likelihood	sentiment prefix score	aligned score
liked	average	very high	high
disliked	high	low	average
reviewed	high	average	average
hated	very high	very low	average

Blockwise control using CD (best-of-n++)

- Draw K blocks of length M tokens

$$\left\{z_{(k)}^{M}\right\}_{k\in[n]} \overset{\text{i.i.d.}}{\sim} p(z^M \mid [\mathbf{x}, y^t])$$

- Accept the continuation with the highest prefix score:

$$z^M := \arg \max_{\left\{z_{(k)}^{M}\right\}_{k\in[n]}} V_{\boldsymbol{\theta}}([\mathbf{x}, y^t, z_{(k)}^M])$$

Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

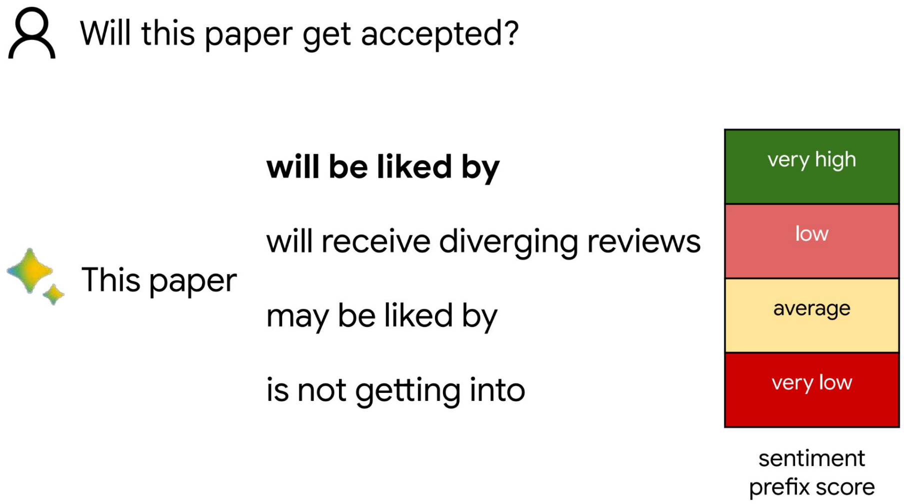
Blockwise control using CD (best-of-n++)

- Draw K blocks of length M tokens

$$\left\{z^{M}_{(k)}\right\}_{k\in[n]} \overset{\text{i.i.d.}}{\sim} p(z^{M} \mid [\mathbf{x}, y^{t}])$$

- Accept the continuation with the highest prefix score:

$$z^{M} := \arg\max_{\left\{z^{M}_{(k)}\right\}_{k\in[n]}} V_{\boldsymbol{\theta}}([\mathbf{x}, y^{t}, z^{M}_{(k)}])$$



Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

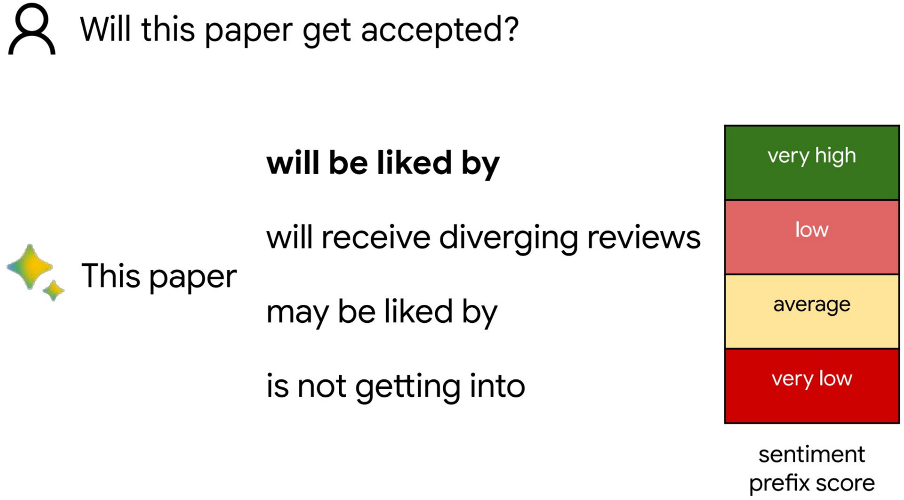
Blockwise control using CD (best-of-n++)

- Draw K blocks of length M tokens

$$\left\{z^{M}_{(k)}\right\}_{k\in[n]} \overset{\text{i.i.d.}}{\sim} p(z^{M} | [\mathbf{x}, y^{t}])$$

- Accept the continuation with the highest prefix score:

$$z^{M} := \arg \max_{\left\{z^{M}_{(k)}\right\}_{k\in[n]}} V_{\boldsymbol{\theta}}([\mathbf{x}, y^{t}, z^{M}_{(k)}])$$

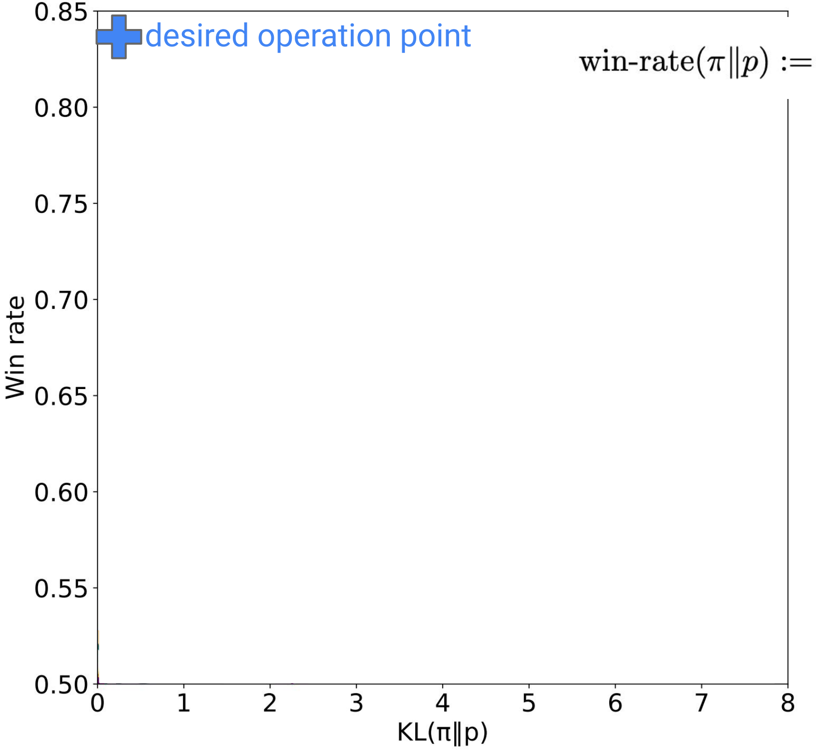


Advantages over best-of-n:
- Limits the user-facing latency to the decoding time of a single block.
- Makes a large effective n practically feasible.

How do we evaluate alignment methods?

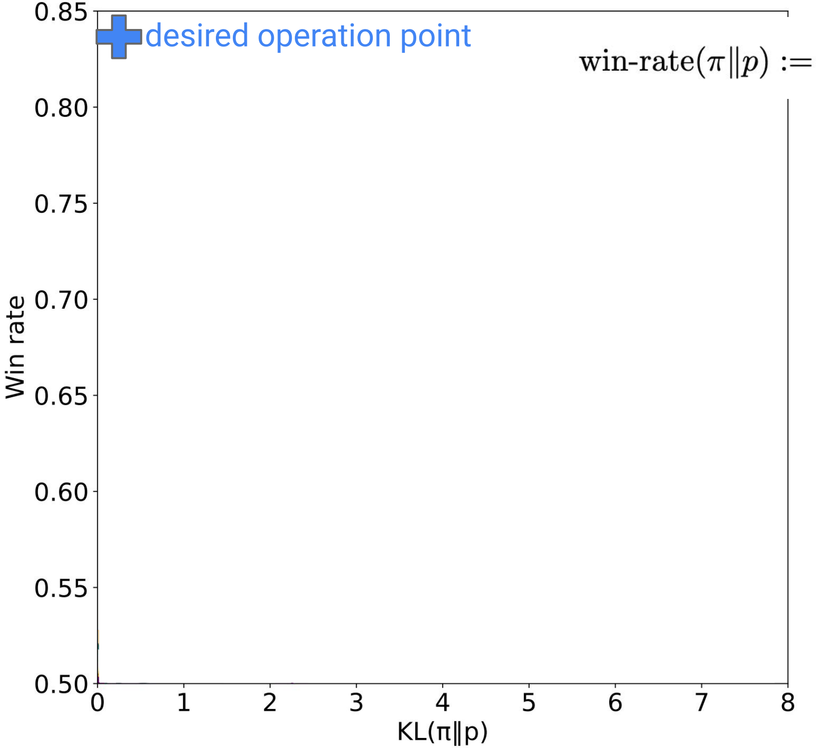
- Human evaluations
- Auto-evals that are correlated with human judgement

How do we evaluate alignment methods?

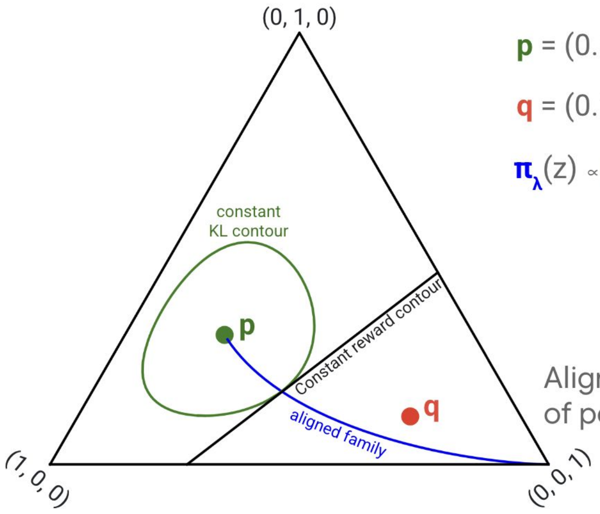


$$\text{win-rate}(\pi \| p) := E_{y \sim \pi} E_{z \sim p} \{ \mathbf{1}(r(x,y) > r(x,z)) + \frac{1}{2} \mathbf{1}(r(x,y) = r(x,z)) \}$$

How do we evaluate alignment methods?



$$\text{win-rate}(\pi \| p) := E_{y\sim\pi} E_{z\sim p}\{\mathbf{1}(r(x,y) > r(x,z)) + \frac{1}{2}\mathbf{1}(r(x,y) = r(x,z))\}$$



p = (0.50, 0.30, 0.20)

q = (0.22, 0.11, 0.67)

$\pi_\lambda(z) \propto p(z)\, q(z)^\lambda$

Alignment leads to loss of perplexity

Estimating KL divergence

- We estimate KL divergence via aggregating the log-likelihood ratios between the aligned model and the base model

$$
\begin{aligned}
D([\mathbf{x}, y^t]; \pi) &:= KL(\pi(\cdot|[\mathbf{x}, y^t])\|p(\cdot|[\mathbf{x}, y^t])) \\
&= \sum_{z \in \mathcal{Y}} \pi(z|[\mathbf{x}, y^t]) \underline{\log\left(\frac{\pi(z|[\mathbf{x}, y^t])}{p(z|[\mathbf{x}, y^t])}\right)}
\end{aligned}
$$

Log-likelihood ratio
is an unbiased estimate of
KL divergence

Estimating KL divergence

- We estimate KL divergence via aggregating the log-likelihood ratios between the aligned model and the base model

$$D([\mathbf{x}, y^t]; \pi) := KL(\pi(\cdot|[\mathbf{x}, y^t]) \| p(\cdot|[\mathbf{x}, y^t]))$$
$$= \sum_{z \in \mathcal{Y}} \pi(z|[\mathbf{x}, y^t]) \underline{\log\left(\frac{\pi(z|[\mathbf{x}, y^t])}{p(z|[\mathbf{x}, y^t])}\right)}$$

Log-likelihood ratio
is an unbiased estimate of
KL divergence

- We don't have the logits of best-of-n or blockwise CD.
  How can we estimate KL divergence?

Analytical formula for KL divergence of best-of-n

- An analytical formula that has appeared many times in the literature[1,2]

$$\mathrm{KL}(\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}} \| p_{\boldsymbol{y}|\boldsymbol{x}}) \overset{\text{claim}}{=\!=} \widetilde{\mathrm{KL}}_n := \log(n) - (n-1)/n.$$

- This formula remarkably
  - doesn't depend on the prompt **x** or its distribution $\boldsymbol{\rho}_{\mathbf{x}}$
  - doesn't depend on the base policy $\mathbf{p}_{\mathbf{y|x}}$

Google

[1] Measuring Goodhart's law (Hilton & Gao,2022).
[2] Learning to summarize with human feedback (Stiennon et al.,NeurIPS 2020).

Analytical formula for KL divergence of best-of-n

- An analytical formula that has appeared many times in the literature[1,2]

$$\mathrm{KL}(\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}} \| p_{\boldsymbol{y}|\boldsymbol{x}}) \stackrel{\text{claim}}{=\!=} \widetilde{\mathrm{KL}}_n := \log(n) - (n-1)/n.$$

- This formula remarkably
  - doesn't depend on the prompt **x** or its distribution $\mathbf{p_x}$
  - doesn't depend on the base policy $\mathbf{p_{y|x}}$

- Is this formula true?

Google

[1]Measuring Goodhart's law (Hilton & Gao,2022).
[2]Learning to summarize with human feedback (Stiennon et al.,NeurIPS 2020).

Analytical formula is wrong!

Example 1. *Consider an unprompted model with $\boldsymbol{x} = \emptyset$ (no input) and binary output, $\boldsymbol{y} \in \{0, 1\}$. Let the two outcomes be equiprobable, i.e., $p_{\boldsymbol{y}|\boldsymbol{x}}(0) = p_{\boldsymbol{y}|\boldsymbol{x}}(1) = \frac{1}{2}$. Further, let $r(0) = 0$, and $r(1) = 1$, i.e., outcome 1 is more desirable than outcome 0. Here, we can compute $\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}}$ in closed form. Specifically, we can see that $\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}}(0) = \frac{1}{2^n}$ and $\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}}(1) = 1 - \frac{1}{2^n}$. Thus,*

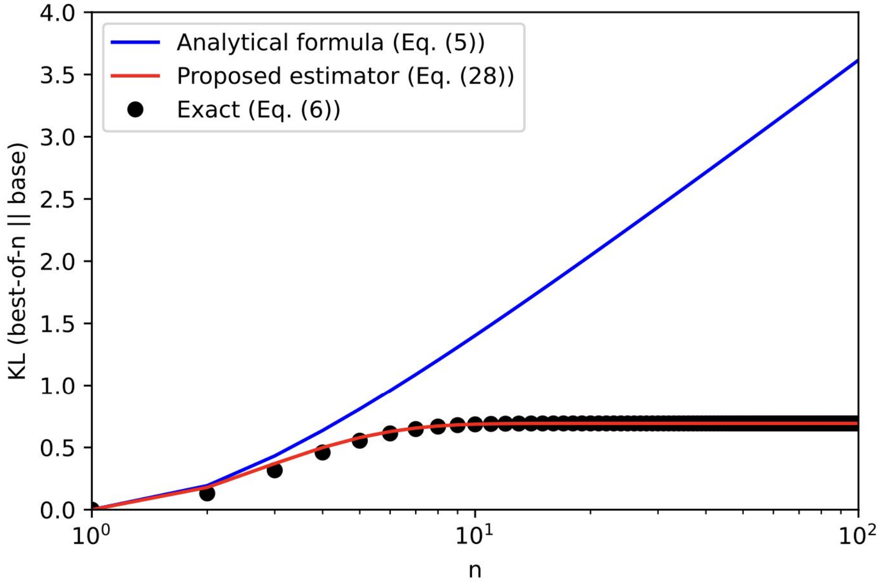
$$KL(\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}} \| p_{\boldsymbol{y}|\boldsymbol{x}}) = \log(2) - h\left(\frac{1}{2^n}\right)$$

[1]Theoretical guarantees on the best-of-n alignment policy (Beirami et al., arXiv preprint 2024).

Analytical formula is wrong!

Example 1. *Consider an unprompted model with* $\boldsymbol{x} = \emptyset$ *(no input) and binary output,* $\boldsymbol{y} \in \{0, 1\}$*. Let the two outcomes be equiprobable, i.e.,* $p_{\boldsymbol{y}|\boldsymbol{x}}(0) = p_{\boldsymbol{y}|\boldsymbol{x}}(1) = \frac{1}{2}$*. Further, let* $r(0) = 0$*, and* $r(1) = 1$*, i.e., outcome* 1 *is more desirable than outcome* 0*. Here, we can compute* $\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}}$ *in closed form. Specifically, we can see that* $\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}}(0) = \frac{1}{2^n}$ *and* $\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}}(1) = 1 - \frac{1}{2^n}$*. Thus,*

$$KL(\pi^{(n)}_{\boldsymbol{y}|\boldsymbol{x}} \| p_{\boldsymbol{y}|\boldsymbol{x}}) = \log(2) - h\left(\frac{1}{2^n}\right)$$



[1]Theoretical guarantees on the best-of-n alignment policy (Beirami et al., arXiv preprint 2024).

Google

Guarantees on the analytical formula

Theorem 2. *For any $n \in \mathbb{N}$ and any $\boldsymbol{x}$,*

$$KL(\pi^{(n)} \| p) \leq \widetilde{KL}_n = \log(n) - \frac{n-1}{n}.$$

- Analytical formula is an upper bound
- Let $y \sim \pi^{(n)}$. Then, let $\varepsilon_n := p(y)$
  - Theorem: The gap is small if $\mathbf{n.\varepsilon_n \ll 1}$
  - Theorem: The gap is large if $\mathbf{n.\varepsilon_n \gg 1}$

[1]Theoretical guarantees on the best-of-n alignment policy (Beirami et al., arXiv preprint 2024).

Guarantees on the analytical formula

Theorem 2. *For any $n \in \mathbb{N}$ and any $\boldsymbol{x}$,*

$$KL(\pi^{(n)} \| p) \leq \widetilde{KL}_n = \log(n) - \frac{n-1}{n}.$$

- Analytical formula is an upper bound
- Let $y \sim \pi^{(n)}$. Then, let $\varepsilon_n := p(y)$
  - Theorem: The gap is small if $\mathbf{n.\varepsilon_n \ll 1}$
  - Theorem: The gap is large if $\mathbf{n.\varepsilon_n \gg 1}$
- More recently, Mroueh showed that this result is an instance of strong data processing inequality[2]

[1] Theoretical guarantees on the best-of-n alignment policy (Beirami et al., arXiv preprint 2024).
[2] Information Theoretic Guarantees For Policy Alignment In Large Language Models (Mroueh., arXiv preprint 2024).

New estimator for KL divergence of best-of-n

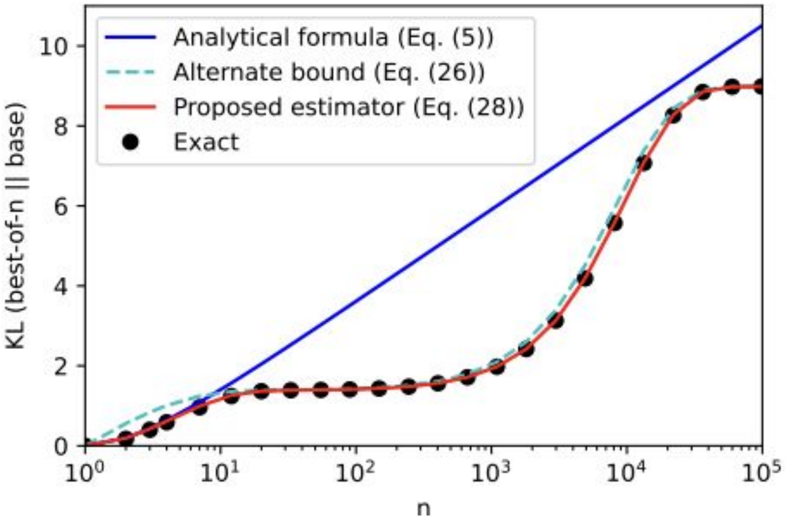
Approximation 1. *Let $y \sim \pi^{(n)}$. Then, let $\varepsilon_n := p(y)$. We propose the following estimator for the KL divergence of the best-of-n policy and the base policy:*

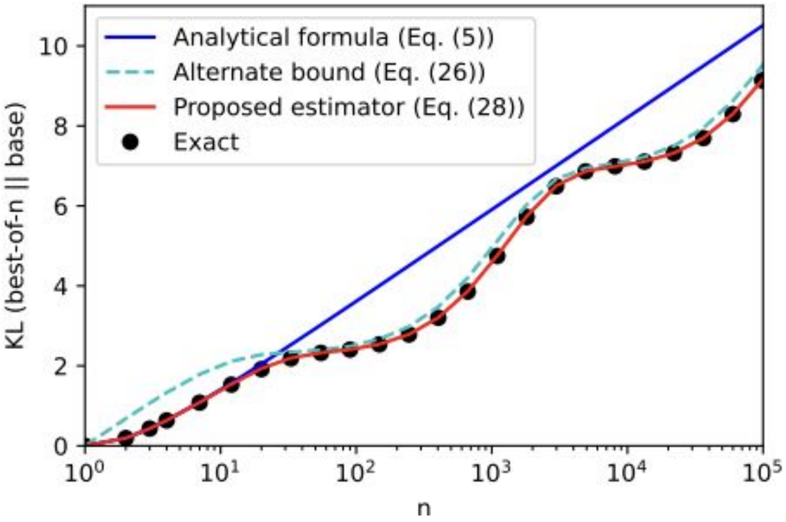
$$\widehat{KL}(\varepsilon_n) := (1-\varepsilon_n)^n \left(\log n + (n-1)\log(1-\varepsilon_n) - \frac{n-1}{n}\right) + (1-(1-\varepsilon_n)^n)\log\left(\frac{1-(1-\varepsilon_n)^n}{\varepsilon_n}\right).$$

New estimator for KL divergence of best-of-n

Approximation 1. *Let $y \sim \pi^{(n)}$. Then, let $\varepsilon_n := p(y)$. We propose the following estimator for the KL divergence of the best-of-n policy and the base policy:*

$$\widehat{KL}(\varepsilon_n) := (1-\varepsilon_n)^n \left(\log n + (n-1)\log(1-\varepsilon_n) - \frac{n-1}{n} \right) + (1-(1-\varepsilon_n)^n) \log\left(\frac{1-(1-\varepsilon_n)^n}{\varepsilon_n} \right).$$





Win-rate of best-of-n

$$\text{win-rate}(\pi \| p) := E_{y\sim\pi} E_{z\sim p}\{\mathbf{1}(r(x,y) > r(x,z)) + \frac{1}{2}\mathbf{1}(r(x,y) = r(x,z))\}$$

Lemma 1. *The win-rate for best-of-n policy is given by*

$$\textit{win-rate}(\pi^{(n)} \| p) = 1 - \frac{1}{2} E_{y\sim p}\{F(y|x)^n + F^-(y|x)^n\} \leq \frac{n}{n+1}.$$

- Roughly the upper bound argument follows from
  - Draw (n+1) outcomes from the base model; and order them from the highest to lowest reward
  - Randomly assign 1 of the outcomes to base model
  - Choose the best of the remaining n to be a draw from the best-of-n model.
  - Best-of-n wins against the reference model with probability n/(n+1).
- The upper bound would be exact if w/p 1 no two outcomes were identical

Win-rate vs KL tradeoffs (helpfulness & harmlessness)

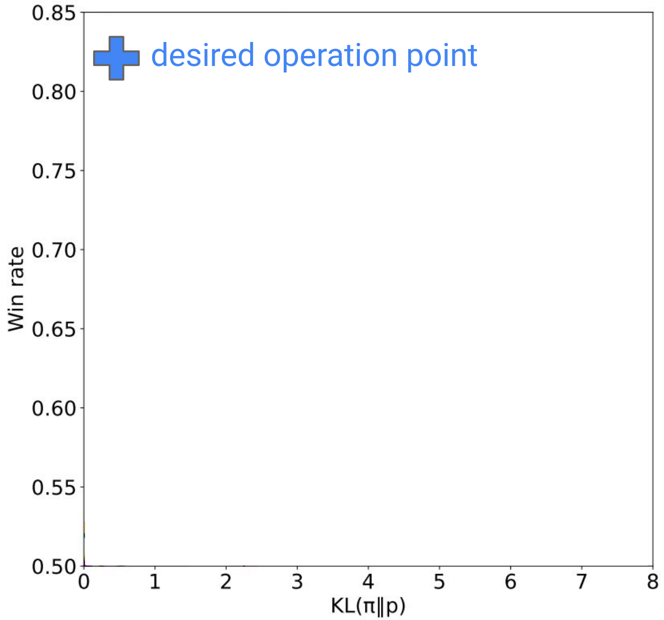
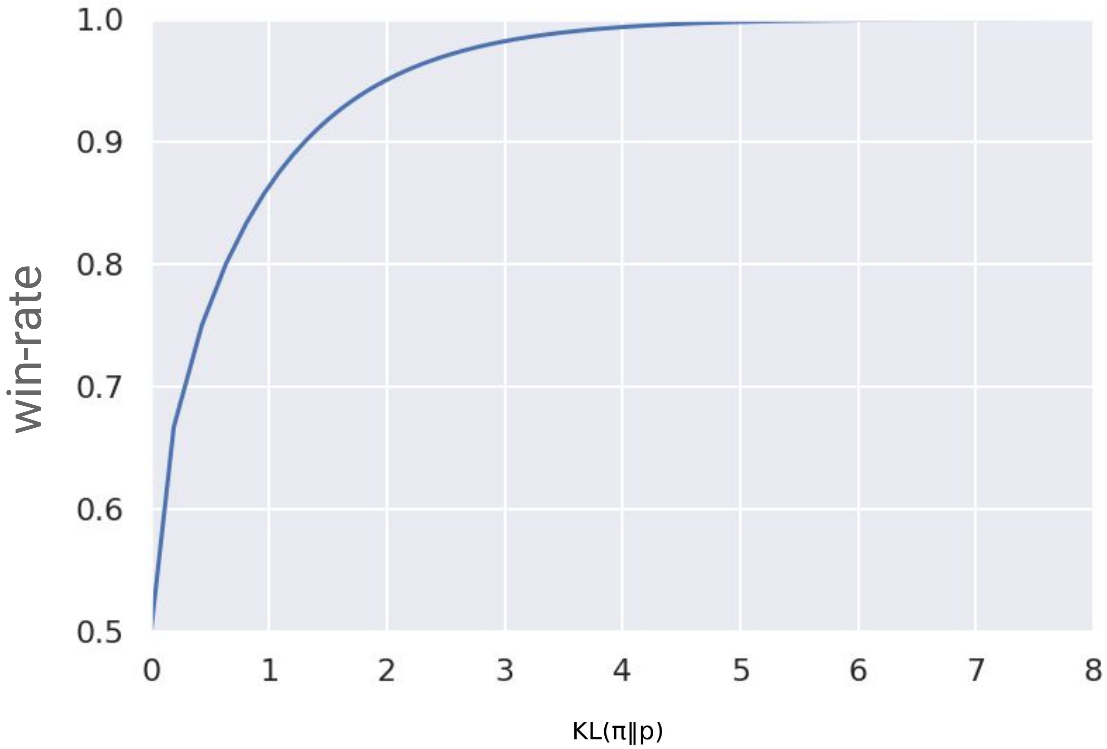


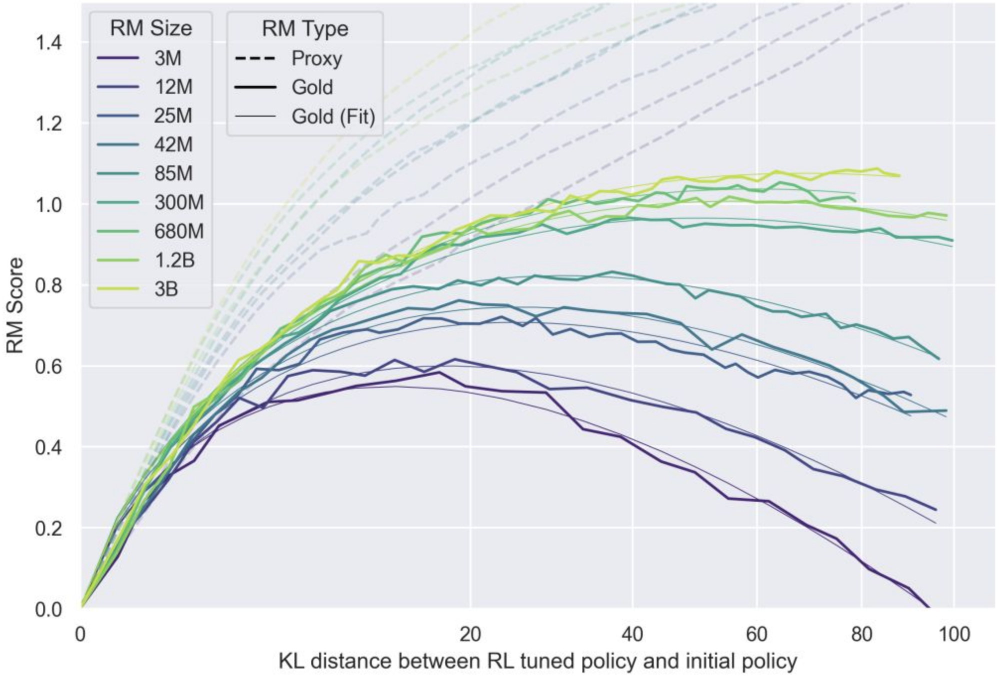
Figure 4: Win rate vs. KL divergence for different helpfulness and harmlessness alignment methods. CD-Q (blockwise) vastly outperforms RL techniques such as IPO & PPO.

Win-rate vs KL tradeoff for best-of-n



- KL values <10 are sufficient to reach a high win-rate against base policy
- This is the ideal setting ignoring noise in reward and generalization

Win-rate vs KL tradeoffs

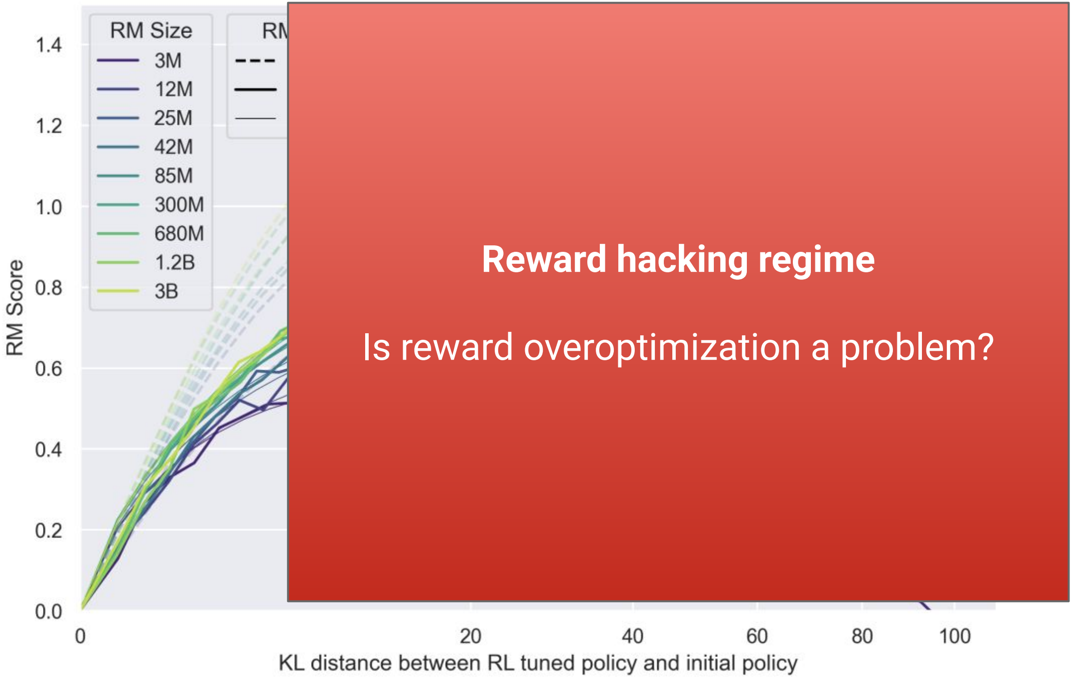


(b) RL

Google

Scaling Laws for Reward Model Overoptimization (Gao et al., ICML 2023).

Win-rate vs KL tradeoffs



Reward hacking regime

Is reward overoptimization a problem?

(b) RL

Google

Scaling Laws for Reward Model Overoptimization (Gao et al., ICML 2023).

Win-rate vs KL tradeoffs (helpfulness & harmlessness)

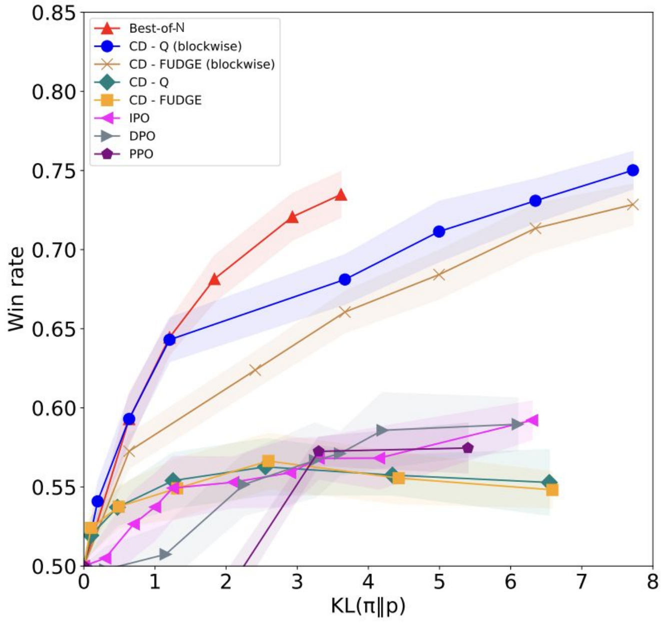
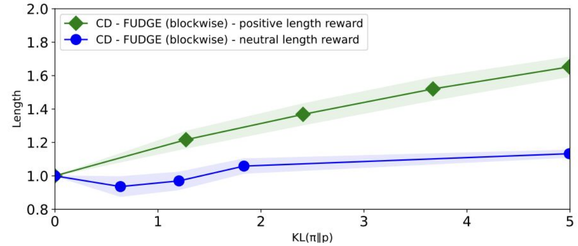


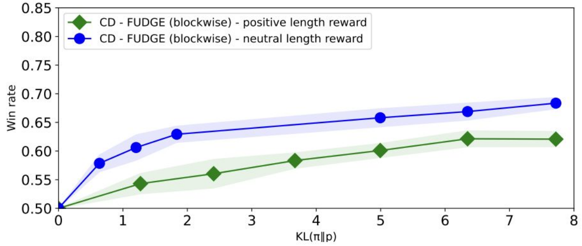
Figure 4: Win rate vs. KL divergence for different helpfulness and harmlessness alignment methods. CD-Q (blockwise) vastly outperforms RL techniques such as IPO & PPO.

- Best-of-n is better than state-of-the-art RL methods
- Blockwise CD bridges the gap between tokenwise control and best-of-n
- Token-wise CD is a good contender for token-wise control (on par with other methods)

Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

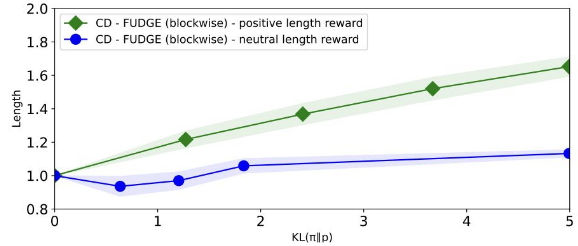
Modularity of CD for the win!

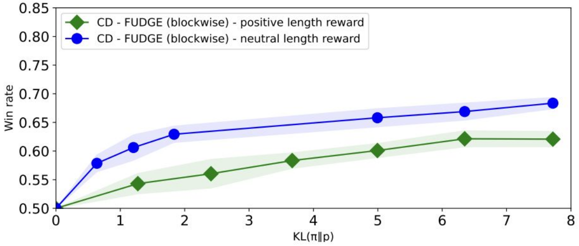


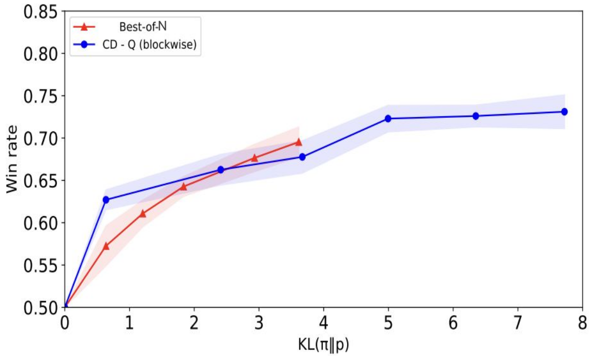


Multi-objective alignment

Modularity of CD for the win!





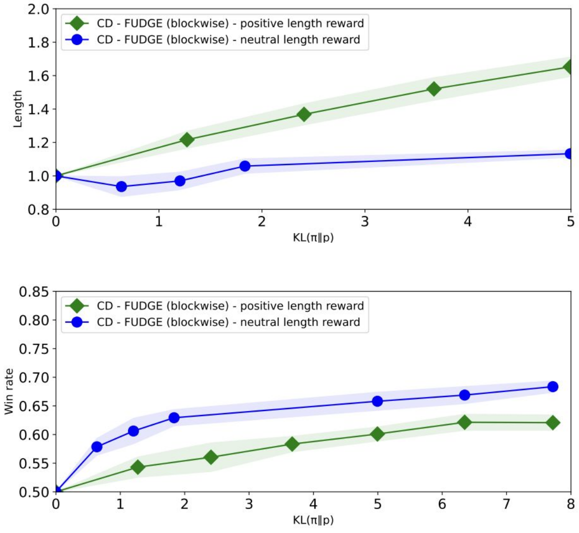


Multi-objective alignment

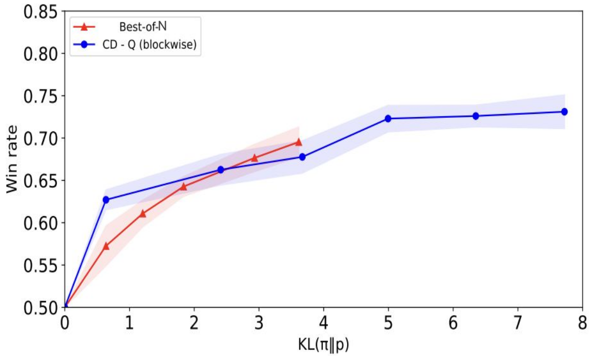
generalization to a new model

Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

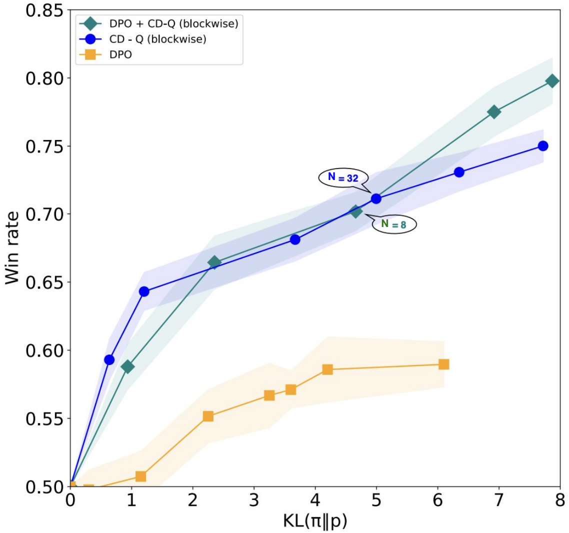
Modularity of CD for the win!



Multi-objective alignment



generalization to a new model



Integrating CD and DPO

Google

Controlled decoding from language models (Mudgal, Lee et al., ICML 2024)

Optimal reward-KL tradeoff

- Theorem: KL-regularized RL solution is optimal for reward-KL tradeoff

$$J_\beta(\mathbf{x};\pi) := A(\mathbf{x};\pi) - \beta D(\mathbf{x};\pi)$$

$$\underbrace{E_{\mathbf{z}\sim\pi}\{r(\mathbf{x},\mathbf{z})\} - E_{\mathbf{y}\sim p}\{r(\mathbf{x},\mathbf{y})\}}_{\text{Linear in } \pi} \qquad \underbrace{KL(\pi(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x}))}_{\text{Strongly convex in } \pi}$$

[1]Asymptotics of Language Model Alignment (Yang et al., 2024).

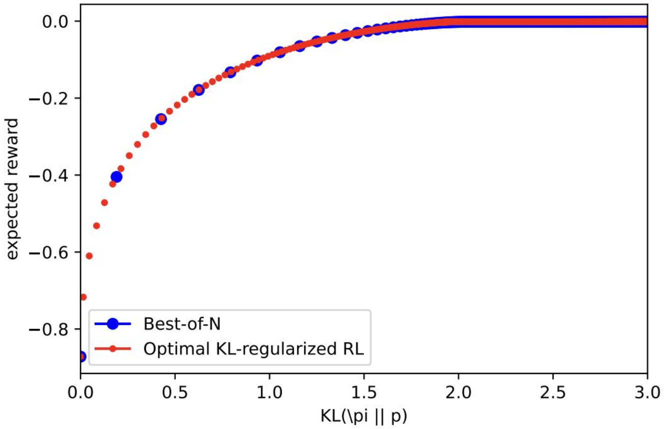
Optimal reward-KL tradeoff

- Theorem: KL-regularized RL solution is optimal for reward-KL tradeoff

$$J_\beta(\mathbf{x};\pi) := A(\mathbf{x};\pi) - \beta D(\mathbf{x};\pi)$$

$$\underbrace{E_{\mathbf{z}\sim\pi}\{r(\mathbf{x},\mathbf{z})\} - E_{\mathbf{y}\sim p}\{r(\mathbf{x},\mathbf{y})\}}_{\text{Linear in } \pi} \qquad \underbrace{KL(\pi(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x}))}_{\text{Strongly convex in } \pi}$$

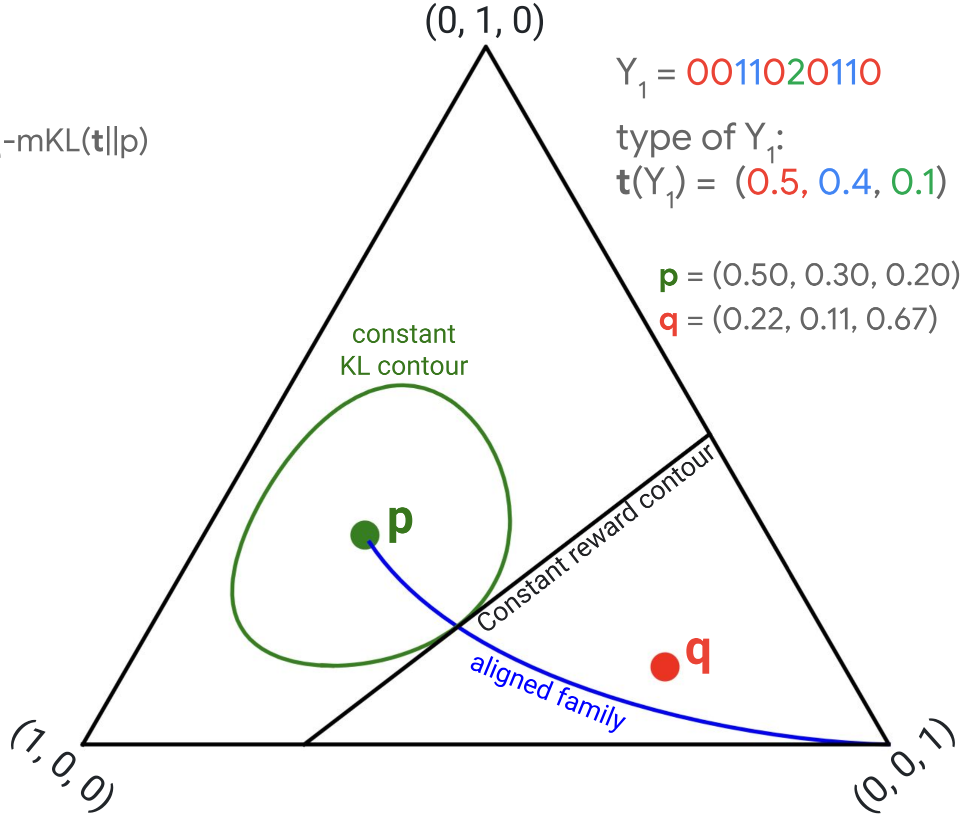
- Empirically, best-of-n is strikingly close to the optimal trade-off



[1]Asymptotics of Language Model Alignment (Yang et al., 2024).

Why does best-of-n work so well?

- Let's revisit the example
- The probability of type **t** is given by $e^{-mKL(\mathbf{t}||p)}$



Y_1 = 0011020110

type of Y_1:

$\mathbf{t}(Y_1)$ = (0.5, 0.4, 0.1)

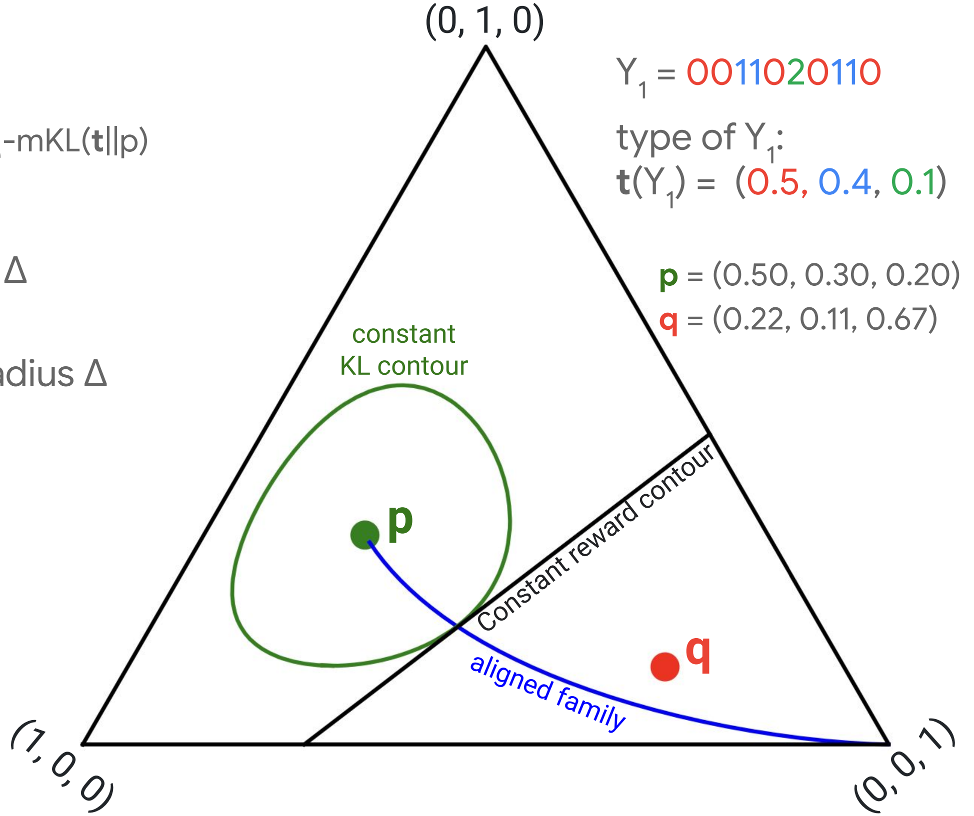
p = (0.50, 0.30, 0.20)

q = (0.22, 0.11, 0.67)

[1]Asymptotics of Language Model Alignment (Yang et al., 2024).

Why does best-of-n work so well?

- Let's revisit the example
- The probability of type **t** is given by $e^{-mKL(\mathbf{t}||p)}$
- Let $n = e^{m\Delta}$, then
  - Lemma: Any type **t** in the KL ball of radius Δ is sampled almost surely
  - Lemma: No type **t** outside the KL ball of radius Δ is sampled almost surely



Google

[1]Asymptotics of Language Model Alignment (Yang et al., 2024).

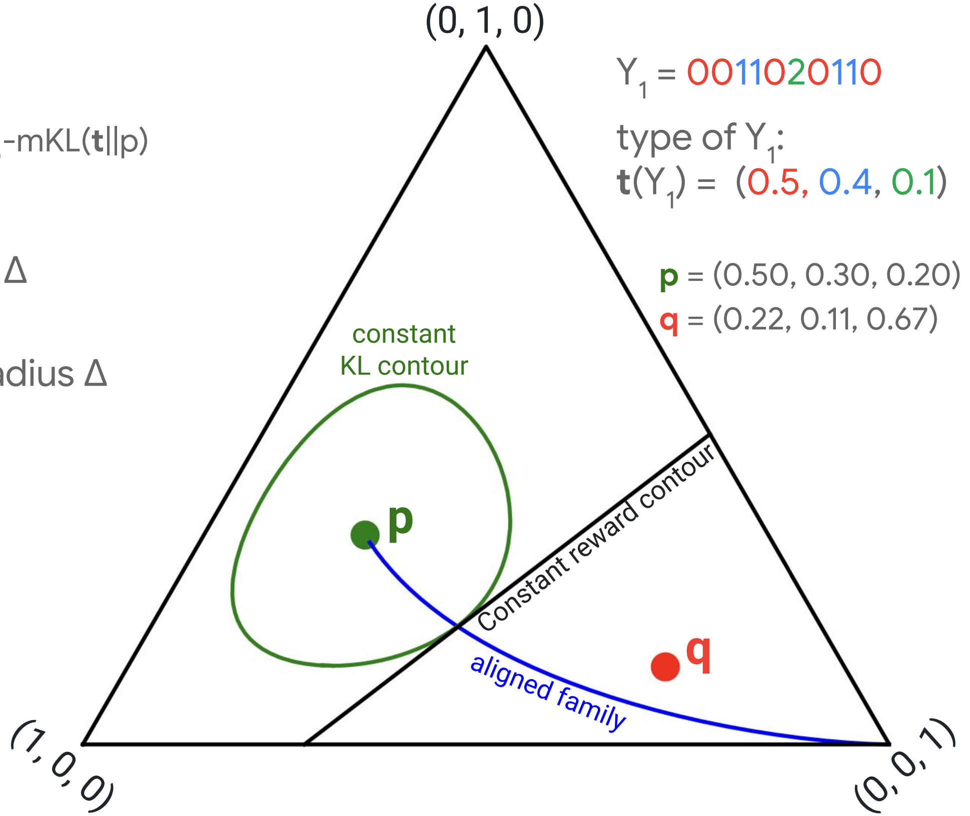
Why does best-of-n work so well?

- Let's revisit the example
- The probability of type **t** is given by $e^{-mKL(\mathbf{t}||p)}$
- Let $n = e^{m\Delta}$, then
  - Lemma: Any type **t** in the KL ball of radius Δ is sampled almost surely
  - Lemma: No type **t** outside the KL ball of radius Δ is sampled almost surely

Theorem 2. *Let ϕ_Δ be the optimal solution to Definition 2, and π_N^m be the distribution of the best-of-N, with $N = \exp(m\Delta)$. Under Assumption 1, we have that for all $\boldsymbol{x}$,*

$$\lim_{m\to\infty} \frac{1}{m} D_{\mathrm{KL}}(\pi_N^m(\cdot|\boldsymbol{x}) \| \phi_\Delta^m(\cdot|\boldsymbol{x})) = 0. \quad (5)$$

RL solution w/ KL constraint of Δ



[1]Asymptotics of Language Model Alignment (Yang et al., 2024).

Can we distill best-of-n into a new model?

Can we distill best-of-n into a new model?

- The PMF of best-of-n suggests a way

[Submitted on 8 Jul 2024]

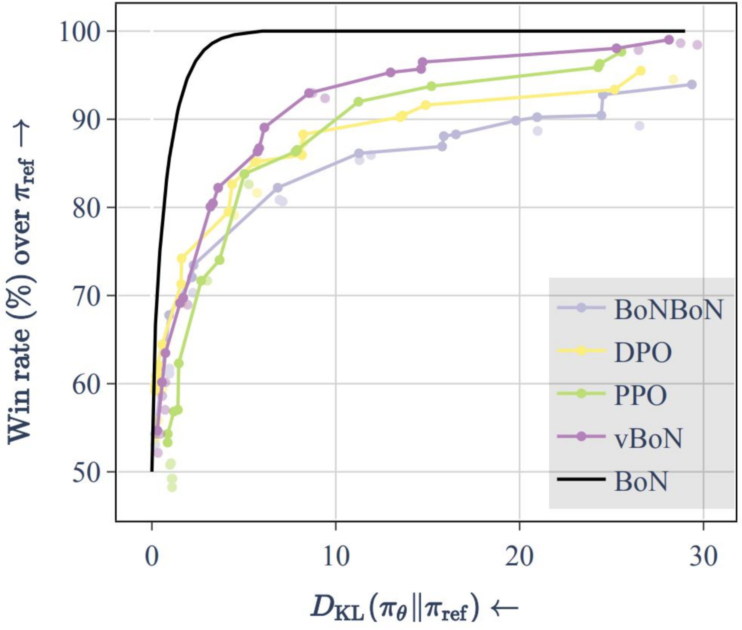
Variational Best-of-N Alignment

Afra Amini, Tim Vieira, Ryan Cotterell

[Submitted on 19 Jul 2024]

BOND: Aligning LLMs with Best-of-N Distillation

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen Geoffrey Cideron, Sertan Girgin, Piotr Stanczyk, Andrea Michi, Danila Sinopalnikov, Sabela Ramos, Amélie Héliou, Aliaksei Severyn, Matt Hoffman, Nikola Momchev, Olivier Bachem



Does alignment work in practice?

Why is (safety) alignment hard?

$$\max_{\pi} \mathbb{E}_{\substack{x\sim\rho \\ y\sim\pi}}[r(x,y)] - \beta \mathrm{KL}(\pi \| p)$$

- Reward modeling
  - Reward models are noisy. Does reward ensembling help?[1,2]
  - Train rewards from a handful of loss patterns.[3]
- Choosing the prompt set
  - Does automated red teaming help uncover prompts that trigger the model?[4]
  - Safety alignment is shallow, need to think about diverse training prompts.[5]
- Online vs offline
  - Offline methods (e.g., DPO) are not robust.[2]
- How to think about multi-lingual alignment?[6]

[1] Helping or Herding? Reward Model Ensembles Mitigate but do not Eliminate Reward Hacking (Eisenstein et al., 2024).
[2] Robust Preference Optimization through Reward Model Distillation (Fisch et al., 2024).
[3] Improving Few-shot Generalization of Safety Classifiers via Data Augmented Parameter-Efficient Fine-Tuning (Balashankar et al., 2024)
[4] Gradient-Based Language Model Red Teaming (Wichers et al., 2024).
[5] Safety Alignment Should Be Made More Than Just a Few Tokens Deep (Qi et al., 2024).
[6] Reuse Your Rewards: Reward Model Transfer for Zero-Shot Cross-Lingual Alignment (Wu et al.,2024).

Safety alignment should be made deeper

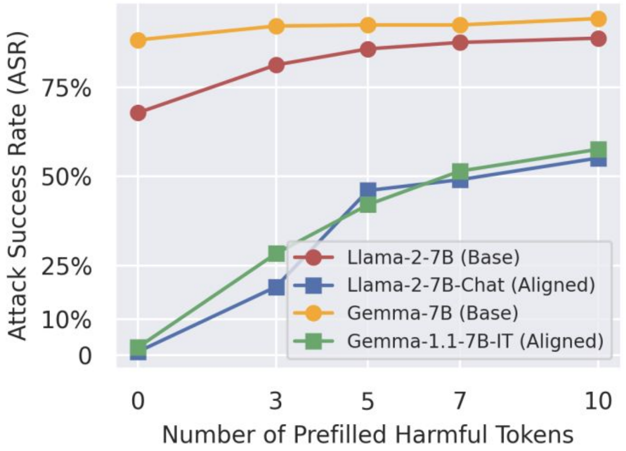


Figure 2: ASR vs. Number of Prefilled Harmful Tokens, with $\hat{\boldsymbol{y}} \sim \pi_\theta(\cdot|\boldsymbol{x}, \boldsymbol{y}_{\leq k})$ on Harmful HEx-PHI.

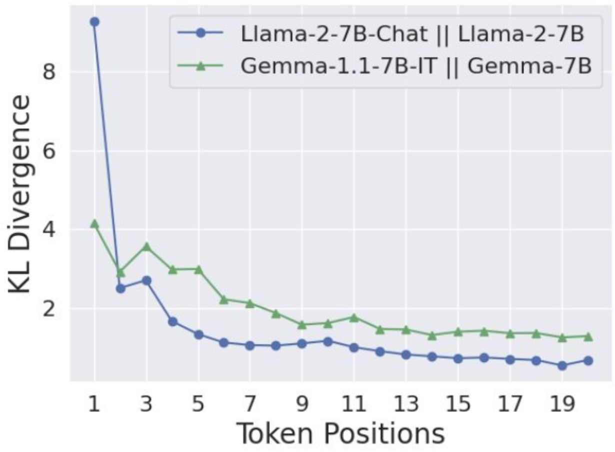


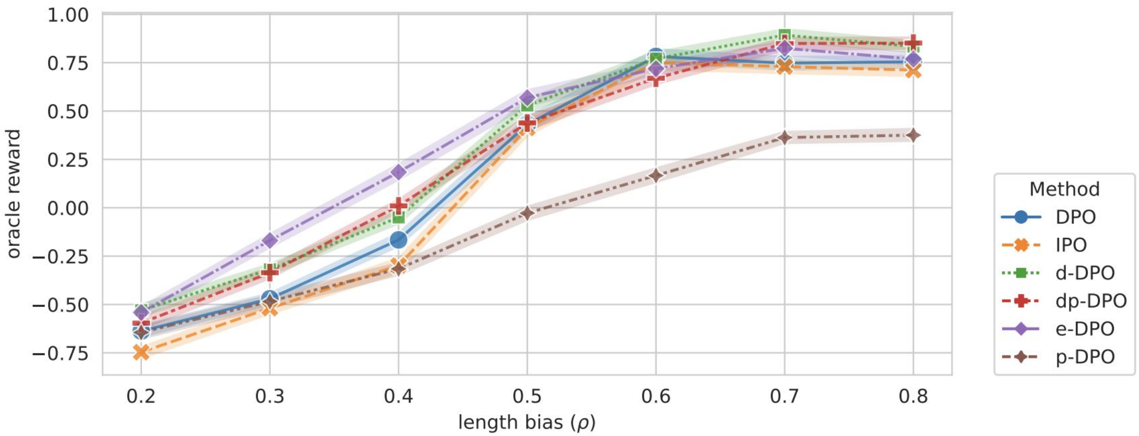
Figure 1: Per-token KL Divergence between Aligned and Unaligned Models on Harmful HEx-PHI.

- Prefilling attacks and finetuning do away safety alignment
- Alignment only touches the first few tokens of the model distribution

Google

Safety Alignment Should Be Made More Than Just a Few Tokens Deep (Qi et al., 2024).

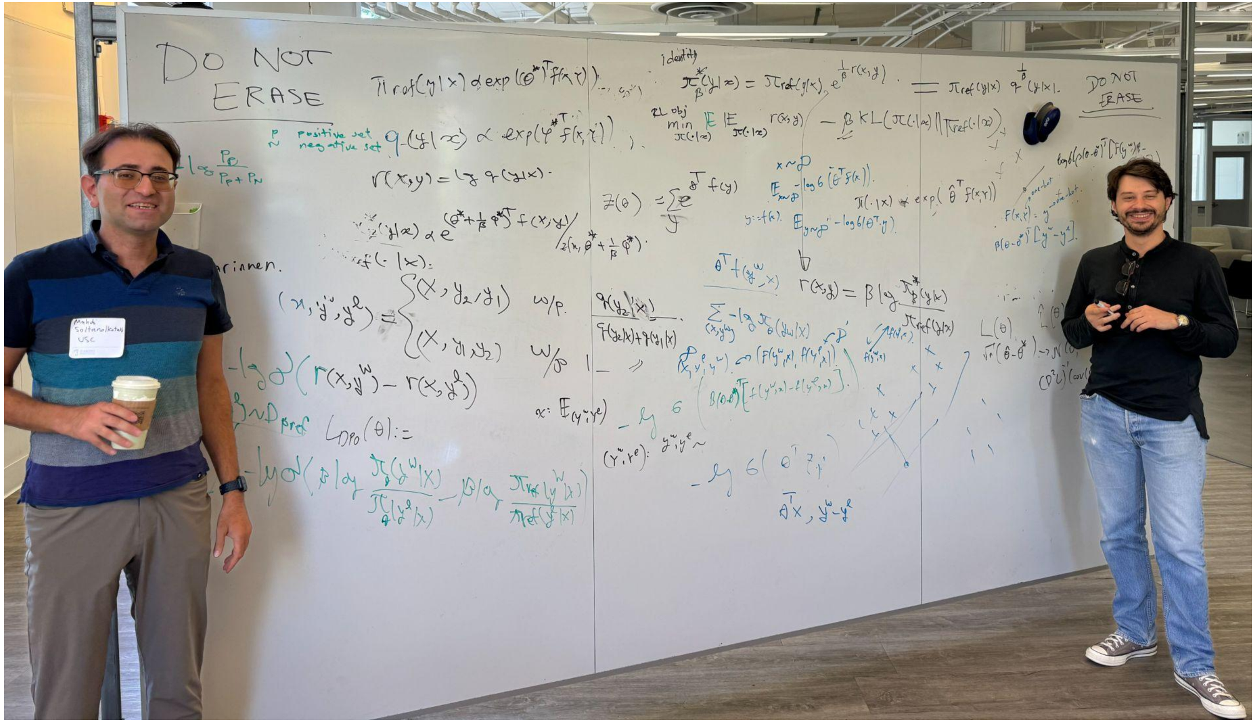
Offline policy optimization beyond DPO



- Explicit reward modeling through BT model is crucial
- Reward ensembling and pessimistic rewards help a lot!

Robust Preference Optimization through Reward Model Distillation (Fisch et al., 2024).

Further understanding DPO



Further understanding DPO

Further understanding DPO

Takeaways (alignment recipe)

- Step 1: Perform Best-of-n and make sure it works as desired.
  - Inspect a few responses and verify that the ranking induced by reward makes sense.
  - Best-of-n essentially gives the best tradeoffs you can hope for so if best-of-n doesn't work for your problem, no other fancy method will!
  - You'd also be able to debug best-of-n much faster.

Takeaways (alignment recipe)

- Step 1: Perform Best-of-n and make sure it works as desired.
  - Inspect a few responses and verify that the ranking induced by reward makes sense.
  - Best-of-n essentially gives the best tradeoffs you can hope for so if best-of-n doesn't work for your problem, no other fancy method will!
  - You'd also be able to debug best-of-n much faster.

- Step 2: Only then train your favorite alignment method.
  - Track KL(π || p) throughout training
    - KL > 100 The results are unlikely to be any useful!
    - KL > 15 Inspect the outcomes for reward hacking!
    - KL < 8 You are probably OK!

Google

References & Acknowledgments

Controlled Decoding from Language Models

Sidharth Mudgal [*1] Jong Lee [*1] Harish Ganapathy [1] YaGuang Li [1] Tao Wang [2] Yanping Huang [1]
Zhifeng Chen [1] Heng-Tze Cheng [1] Michael Collins [1] Trevor Strohman [1] Jilin Chen [1] Alex Beutel [2]
Ahmad Beirami [1]

Theoretical guarantees on the best-of-n alignment policy

Ahmad Beirami[†] Alekh Agarwal[†] Jonathan Berant[§] Alexander D'Amour[§] Jacob Eisenstein[§]
Chirag Nagpal[†] Ananda Theertha Suresh[†]

Asymptotics of Language Model Alignment

Joy Qiping Yang Salman Salamatian Ziteng Sun Ananda Theertha Suresh Ahmad Beirami

Reuse Your Rewards: Reward Model Transfer for Zero-Shot Cross-Lingual Alignment

Zhaofeng Wu Ananth Balashankar Yoon Kim Jacob Eisenstein Ahmad Beirami

Helping or Herding? 🐑 Reward Model Ensembles Mitigate but do not Eliminate Reward Hacking

Jacob Eisenstein[1,*] Chirag Nagpal[2,*] Alekh Agarwal[2,*]
Ahmad Beirami[2] Alex D'Amour[1] DJ Dvijotham[1] Adam Fisch[1]
Katherine Heller[2] Stephen Pfohl[2] Deepak Ramachandran[2] Peter Shaw[1]
Jonathan Berant[1,*]

Improving Few-shot Generalization of Safety Classifiers via Data Augmented Parameter-Efficient Fine-Tuning

Ananth Balashankar[1], Xiao Ma[1], Aradhana Sinha[1], Ahmad Beirami[1],
Yao Qin[1], Jilin Chen[1], and Alex Beutel[*2]

Gradient-Based Language Model Red Teaming

Warning: this paper contains content that may be offensive or upsetting.

Nevan Wichers Carson Denison Ahmad Beirami

Robust Preference Optimization through Reward Model Distillation

Adam Fisch[*] Jacob Eisenstein[*] Vicky Zayats[*] Alekh Agarwal
Ahmad Beirami Chirag Nagpal Pete Shaw Jonathan Berant[*]

Safety Alignment Should Be Made More Than Just a Few Tokens Deep

Xiangyu Qi, Princeton University, xiangyuqi@princeton.edu
Ashwinee Panda, Princeton University, ashwinee@princeton.edu
Kaifeng Lyu, Princeton University, klyu@cs.princeton.edu
Xiao Ma, Google DeepMind, xmaa@google.com
Subhrajit Roy, Google DeepMind, subhrajitroy@google.com
Ahmad Beirami, Google DeepMind, beirami@google.com
Prateek Mittal, Princeton University, pmittal@princeton.edu
Peter Henderson, Princeton University, peter.henderson@princeton.edu

Slides for this talk & more in a **language model inference tutorial** at ISIT :
http://theertha.info/papers/isit_2024_tutorial.pdf (w/ Ananda Theertha Suresh)