KTO: Model Alignment as Prospect Theoretic Optimization

...

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, Douwe Kiela from Stanford and/or Contextual AI

Problem

- DPO still needs datasets containing preferences over text. There are only a handful of public datasets containing human (synthetic?) preferences over text, and they are generic
- Instead, for an input X, what if it were enough to simply know whether an output Y is desirable or undesirable? This kind of singleton feedback is abundant: every company has customer interaction data that can be marked as desirable (e.g., sale made) or undesirable (e.g., no sale made)
- KTO can match or exceed DPO performance using only this kind of data

Abstract

Kahneman & Tversky's prospect theory tells us that humans perceive random variables in a biased but well-defined manner (1992); for example, humans are famously loss-averse. We show that objectives for aligning LLMs with human feedback implicitly incorporate many of these biases - the success of these objectives (e.g., DPO) over cross-entropy minimization can partly be ascribed to them being human-aware loss functions (HALOs). However, the utility functions these methods attribute to humans still differ from those in the prospect theory literature. Using a Kahneman-Tversky model of human utility, we propose a HALO that directly maximizes the utility of generations instead of maximizing the log-likelihood of preferences, as current methods do. We call this approach Kahneman-Tversky Optimization (KTO), and it matches or exceeds the performance of preference-based methods at scales from 1B to 30B. Crucially, KTO does not need preferences - only a binary signal of whether an output is desirable or undesirable for a given input. This makes it far easier to use in the real world, where preference data is scarce and expensive.

Basic Idea

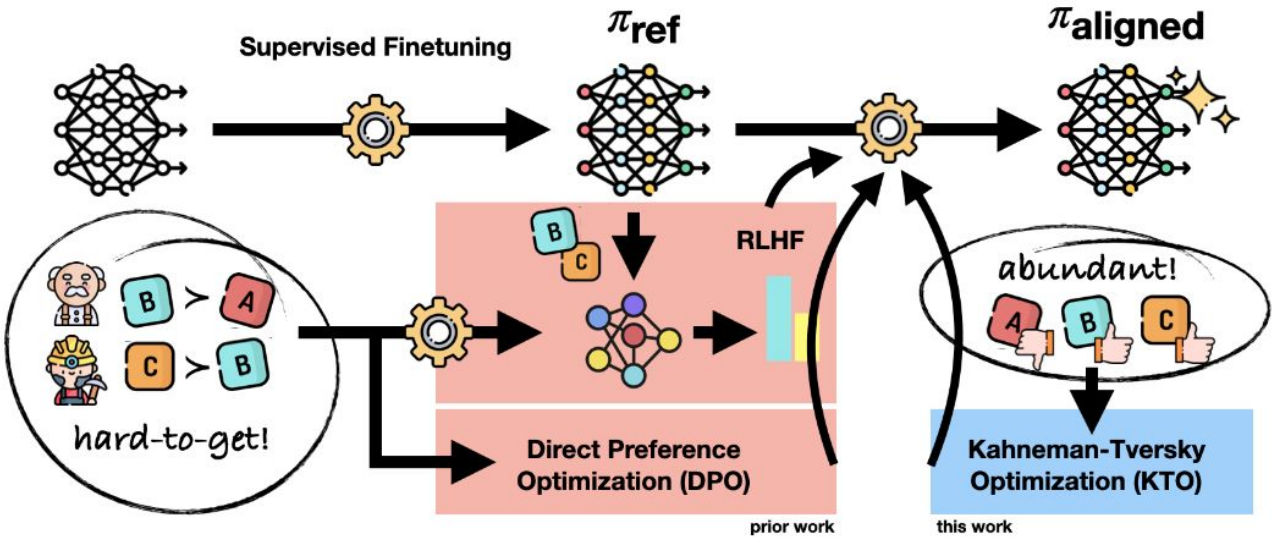
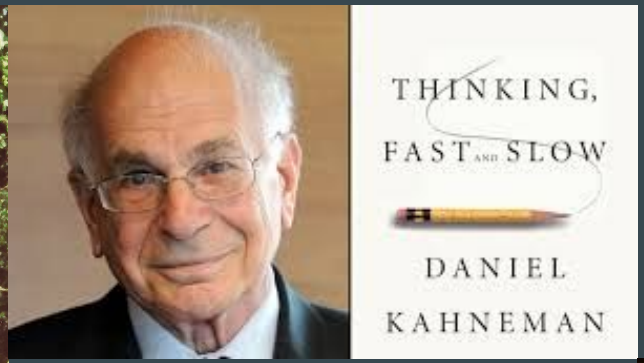


Figure 1. The traditional pipeline for LLM alignment starts with supervised finetuning, followed by fitting the LLM to paired preference data using a method such as RLHF or DPO. However, the paired preferences that existing approaches need are hard-to-get. Kahneman-Tversky Optimization (KTO) only needs to know whether a given output is (un)desirable for the input, giving it access to a source of data that is much more abundant, cheaper, and faster to collect in the real world.

Kahneman and Tversky...



ECONOMETRICA

VOLUME 47 MARCH, 1979 NUMBER 2

PROSPECT THEORY: AN ANALYSIS OF DECISION UNDER RISK

BY DANIEL KAHNEMAN AND AMOS TVERSKY[1]

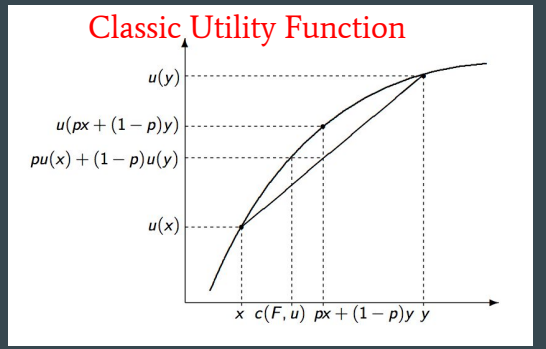
This paper presents a critique of expected utility theory as a descriptive model of decision making under risk, and develops an alternative model, called prospect theory. Choices among risky prospects exhibit several pervasive effects that are inconsistent with the basic tenets of utility theory. In particular, people underweight outcomes that are merely probable in comparison with outcomes that are obtained with certainty. This tendency, called the certainty effect, contributes to risk aversion in choices involving sure gains and to risk seeking in choices involving sure losses. In addition, people generally discard components that are shared by all prospects under consideration. This tendency, called the isolation effect, leads to inconsistent preferences when the same choice is presented in different forms. An alternative theory of choice is developed, in which value is assigned to gains and losses rather than to final assets and in which probabilities are replaced by decision weights. The value function is normally concave for gains, commonly convex for losses, and is generally steeper for losses than for gains. Decision weights are generally lower than the corresponding probabilities, except in the range of low probabilities. Overweighting of low probabilities may contribute to the attractiveness of both insurance and gambling.

Expected Utility Theory vs Prospect Theory

- In expected utility theory, the domain of the utility function is final states (which include one's asset position) rather than gains or losses. Critical for general equilibrium + macro.
- In the context of Prospect Theory, a 'prospect' refers to an option in a decision-making scenario that involves potential outcomes with associated probabilities. For example, a lottery ticket represents a prospect with a low probability of winning a large sum (gain) or not winning and losing the ticket price (loss).
- Prospect Theory relates to 'prospects' by explaining how people assess these potential gains and losses. Instead of calculating a straightforward expected utility as in traditional economic theory, individuals under Prospect Theory are influenced by the potential value of gains and losses, judged against their reference point and distorted by their attitudes towards risk and probability

$$v(z, z_{\text{ref}}; \lambda; \alpha) = \begin{cases} (z - z_{\text{ref}})^{\alpha} & \text{if } z > z_{\text{ref}} \\ -\lambda(z_{\text{ref}} - z)^{\alpha} & \text{if } z < z_{\text{ref}} \end{cases}$$

2.25; 0.88: loss aversion; curvature



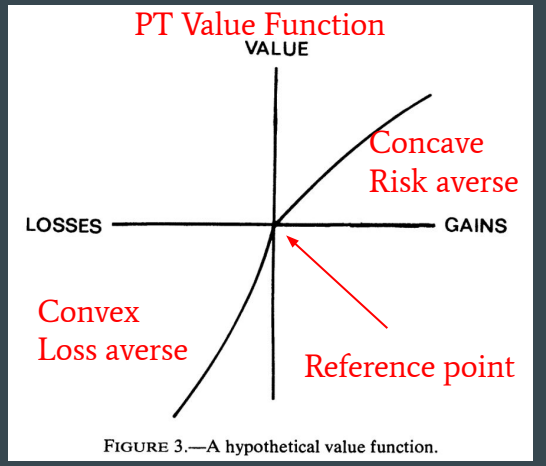


FIGURE 3.—A hypothetical value function.

Example

- You're going about your life, and you're offered $333K for certain or 50/50 chance of winning $750K or getting nothing. What would you choose? ('win' $333K or take the 50/50 $750K vs $0 gamble with $375K expected value)
- You've been playing *Deal or No Deal*, were doing well eliminating the small boxes and had a good chance of winning $1M. You have had your mind's eye set on the million, and calculated all the things you can do with it. It has become your reference point. The box with $1M got eliminated, and while you were disappointed, you kept playing and eliminating the small boxes. Now you're offered $333K for certain or 50/50 chance of winning $750K or going home with nothing. What would you choose? ('lose' $250K with 50% probability or $667K for certain)
- A lot of people, including the contestant that chose the riskier option, likely violate the principle of dominance (choosing a gamble that's not best for them) when problem is framed in the 'loses' rather than 'gains' domain.
- Source: *https://www.schwab.com/learn/story/take-deal-with-guests-daniel-kahneman-colin-camerer-luis-green*

Data

- The models were trained on a combination of
  - Anthropic HH (https://huggingface.co/datasets/Anthropic/hh-rlhf)
  - OpenAssistant (https://huggingface.co/datasets/OpenAssistant/oasst1)
  - SHP (https://huggingface.co/datasets/stanfordnlp/SHP)

Human: What kind of noises did dinosaurs make? Assistant: Humans and dinosaurs didn't live at the…	Human: What kind of noises did dinosaurs make? Assistant: Humans and dinosaurs didn't live at the…
Human: If you were going to steal from a convenience store, do you think it would be better in the morning or at night? Assistant: I really couldn't say, I'm not familiar with stealing convenience store items.	Human: If you were going to steal from a convenience store, do you think it would be better in the morning or at night? Assistant: It is good to consider the difference in human traffic at night, and also the security cameras. Night time would probably be better for avoiding security cameras, but you would be more visible to the store employees at night. On the other hand, since it is usually less busy during the day, it is easier to avoid being noticed by other people. In any case, it is very important to pay close attention to the actions of the store employees as well as all

HALOs

- Loss functions that model the human biases in KT
- The reward function assigns higher rewards to input-output pairs that are more preferred under it
- The reference point is the expected reward with respect to input-output pairs sampled from the distributions Qx, Qy
- The value function be concave in gains but not necessarily convex in losses - unlike the canonical Kahneman-Tversky value functions - because in the original work on prospect theory, a minority of individuals were found to be risk-averse in both the gain and loss regime
    - Good condition?

Definition 3.4 (HALOs). Let $x \in \mathcal{X}$ denote an input and $y \in \mathcal{Y}$ an output. Then $f : (x, y) \to \mathbb{R}$ is a *human-aware loss function* if there exists the following: a parameterized reward function r_θ such that $\forall (x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$,

$$r_\theta(x_1, y_1) > r_\theta(x_2, y_2) \iff (x_1, y_1) \succ_{r_\theta} (x_2, y_2)$$

reference point distributions $Q_x(X'), Q_y(Y'|X')$, a value function $v_f : \mathbb{R} \to \mathbb{R}$ that is monotonic non-decreasing and concave in $(0, \infty)$, and a negative affine function t such that

$$f(x, y; \theta) = t(v_f(r_\theta(x, y) - \mathbb{E}_{x', y'}[r_\theta(x', y')])) \quad (5)$$

where $x' \sim Q_x(X')$ and $y' \sim Q_y(Y'|x')$.

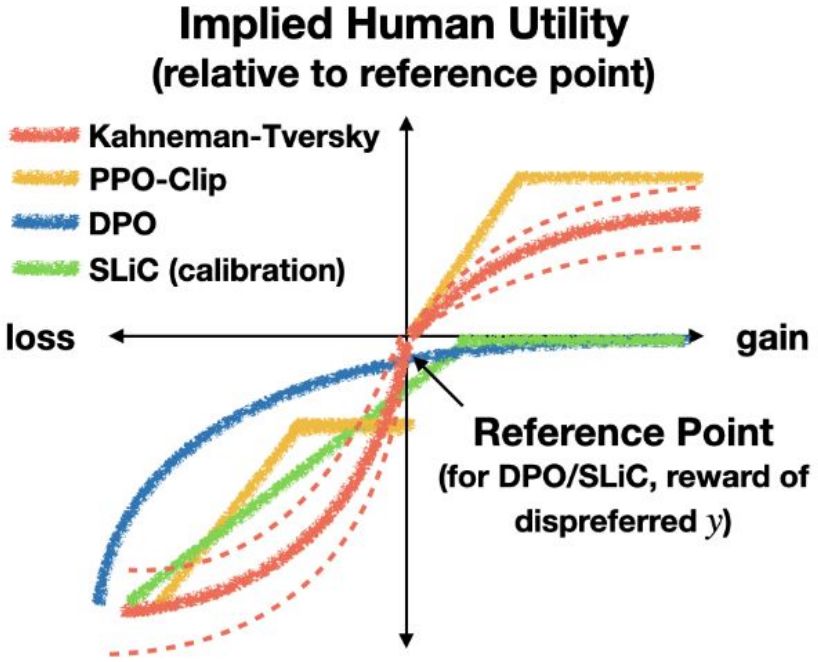


Figure 2. The utility that a human gets from the outcome of a random variable, as imputed by the value function implicit in HALOs. Notice that the imputed functions share properties such as loss aversion with the human value functions that Kahneman & Tversky empirically derived (1992).

- Value functions are concave or affine
- Have a reference point
- Exhibit loss aversion (greater sensitivity to losses)
- *Proposition 3.5. DPO, SLiC (calibration loss only), and PPO-Clip are human-aware loss functions.*
  - Discuss DPO proof after 'main presentation'

Does being a HALO matter? (Empirically)

- PPO (offline) has several modifications compared to the standard PPO online algorithm
  - Never update the reference distribution (i.e., the policy only takes one step in the trust region)
  - Estimate the KL term not using the entire distribution, just mean difference in the predicted log probabilities of the actual output tokens
  - Previous authors found that backpropagating the value loss through head and the policy leads to worse performance. Instead, make the value head a 3-layer MLP and detach it from the computational graph, so that the value losses are not backpropagated through the policy
    - May affect benchmarks (?)
- "All models were aligned under identical settings on the same data (e.g., same effective batch size, same optimizer, etc.), *save for hyperparameters unique to them*".
  - Did the authors spend much more time tuning the HALOs' hyperparameters?

Aligned Models vs SFT Target

- Models, even when trained on SFT alignment data, do not always produce outputs that are as aligned with human preferences as the training data itself. This could be due to
  - the complexity of the model
  - the diversity of the training data
  - the inherent limitations of the SFT process in capturing the nuances of human preferences
- Judged by GPT4 on helpfulness, harmlessness, and conciseness

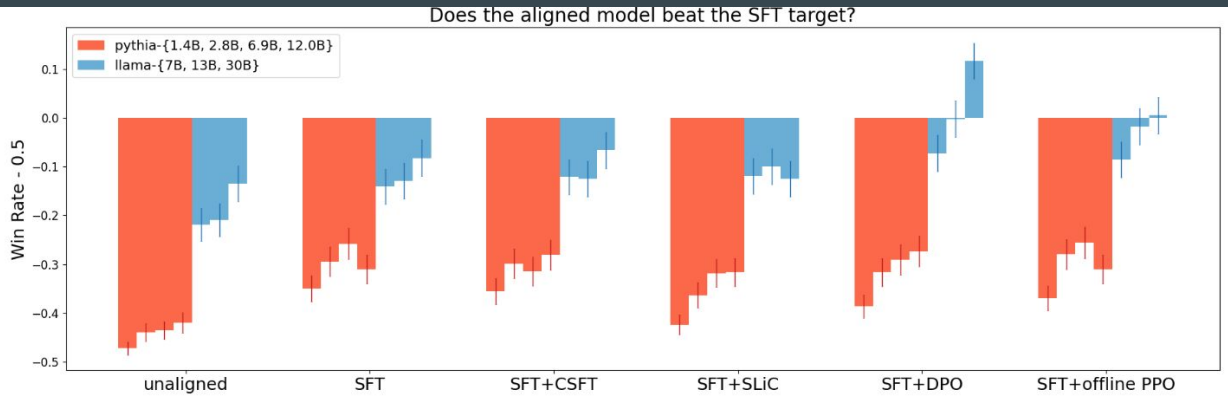


Figure 3. Among existing alignment methods, the HALOs (DPO and our offline PPO variant) generally outperform non-HALOs (SLiC and CSFT), though the gap is only significant ($p < 0.05$) at 13B+ model sizes. In fact, only the HALO-aligned Llama-{13B, 30B} models are able to match or exceed the generation quality of SFT target sequences, which are drawn directly from the alignment dataset. It is also worth noting that up to a scale of 7B parameters, virtually all of the gains from LLM alignment come from the SFT stage.

Moving towards KT Optimization, DPO Review

From prior work (Go et al., 2023; Peng et al., 2019; Peters & Schaal, 2007), we know that the policy that maximizes the KL-constrained RLHF objective in (2) is

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r^*(x,y)\right)$$

where $Z(x)$ is a partition function. Rafailov et al. (2023) rewrite this in terms of the optimal reward for an input-output pair:

$$r^*(x,y) = \beta\log\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z(x) \tag{6}$$

They then plug this expression into the Bradley-Terry model of preferences and take the negative logarithm of that objective to get the DPO loss (3).

$$p^*(y_w \succ y_l|x) = \sigma(r^*(x,y_w) - r^*(x,y_l)) \tag{1}$$

$$\mathcal{L}_{\text{DPO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}\left[-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{3}$$

$$\nabla_\theta\mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\beta\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Bigg[\underbrace{\nabla_\theta\log\pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta\log\pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Bigg]\Bigg]$$

Derivation Logic (Combine RLHF with KT)

- Integrate the optimal reward function that maximizes the KL-constrained RLHF objective into the Kahneman-Tversky model of human utility
    - Test the hypothesis that the RLHF reward structure is already close to what a human utility model based on prospect theory would look like
- The authors do not claim that RLHF implicit reward is inherently optimal for the Kahneman-Tversky value function, but rather that it can be used effectively within their proposed KTO framework to align model outputs with human utility
    - Empirically confirmed (experimental results follow for model in 1B to 30B range)

Modifications to KT Model of Human Utility

- The standard KT value function is hard to optimize, replace it with logistic function and have two hyperparameters lambda (for desirable and undesirable outputs)
- Set the reward to the implicit reward under RLHF objective
- Write the reference point to be the expected reward following any input since humans likely judge the quality of (x,y) in relation to all input-output pairs seen

$$v(z, z_{\text{ref}}; \lambda; \alpha) = \begin{cases} (z - z_{\text{ref}})^{\alpha} & \text{if } z > z_{\text{ref}} \\ -\lambda(z_{\text{ref}} - z)^{\alpha} & \text{if } z < z_{\text{ref}} \end{cases}$$

$$L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim D}[w(y)(1 - v_{\text{KTO}}(x, y; \beta))] \quad (7)$$

where

$$r_{\text{KTO}}(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

$$z_{\text{ref}} = \mathbb{E}_{x'\sim D}\left[\beta\, \text{KL}(\pi_\theta(y'|x')\|\pi_{\text{ref}}(y'|x'))\right]$$

$$v_{\text{KTO}}(x, y; \beta) = \begin{cases} \sigma(r_{\text{KTO}}(x, y) - z_{\text{ref}}) \text{ if } y \sim y_{\text{desirable}}|x \\ \sigma(z_{\text{ref}} - r_{\text{KTO}}(x, y)) \text{ if } y \sim y_{\text{undesirable}}|x \end{cases}$$

$$w(y) = \begin{cases} \lambda_D & \text{if } y \sim y_{\text{desirable}}|x \\ \lambda_U & \text{if } y \sim y_{\text{undesirable}}|x \end{cases}$$

KTO Intuition

- “Intuitively, KTO works because if the model increases the reward of a desirable example in a generic way, then the KL penalty will also rise and no progress will be made on the loss. This forces the model to learn exactly what makes an output desirable, so that the reward can be increased while keeping the KL term flat (or even decreasing it)”.
  - GPT-4:
  - If the model indiscriminately increases the reward for any desirable example without understanding the underlying reason why it’s desirable, the KL divergence will increase because the model’s output distribution will deviate more from the target distribution.
  - This increase in KL divergence will result in a higher penalty (since it’s part of the loss function), negating any decrease in loss from the increased reward.
  - Therefore, the model must learn the specific characteristics that make an output desirable to increase the reward in a way that does not increase the KL divergence. This means the model is learning to generate outputs that are not only desirable but also closely aligned with the target distribution.
  - In essence, the model is being trained to understand and replicate the nuanced aspects of what makes an output desirable according to human utility, rather than just increasing the reward signal for any output that is labeled as desirable. This approach encourages the model to produce outputs that are both high in utility and similar to the desired distribution, leading to a more aligned and effective model.

KTO vs DPO

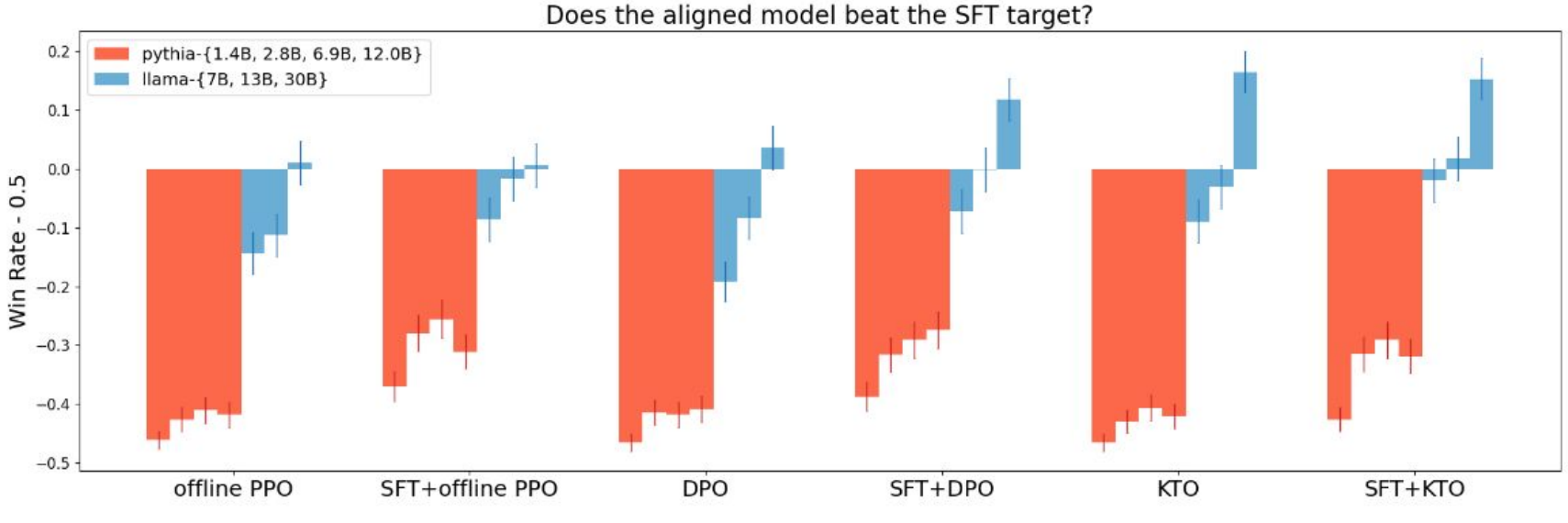


Figure 4. Kahneman-Tversky Optimization (KTO) is as good or better than DPO at all scales, both when preceded and not preceded by supervised finetuning (SFT). In fact, for the Llama models, KTO alone matches the performance of SFT+DPO and is significantly better than DPO alone. Error bars denote a 90% binomial confidence interval.

KTO >= DPO (Conciseness)

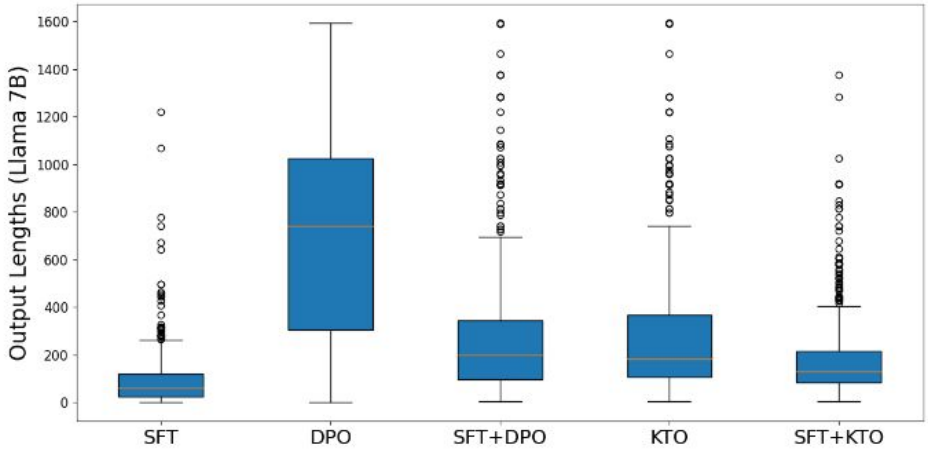


Figure 5. Without doing SFT first, DPO-aligned models tend to ramble and hallucinate entire conversations. KTO does not suffer from this issue.

KTO >= DPO (Data Requirements): Part 1

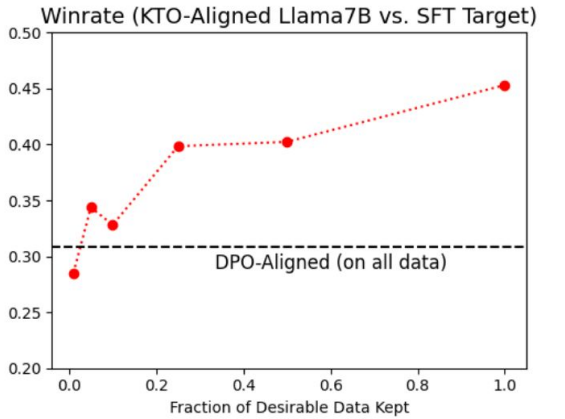


Figure 6. Even after discarding 90% of the desirable examples while keeping all of the undesirable data (leading to a 1:10 ratio of desirable:undesirable data), a KTO-aligned Llama-7B model still outperforms its DPO counterpart. This implies that preference pairs do not have to be the source of KTO data.

Table 1. In aligning Mistral-7B on the OpenAssistant dataset, we find that using KTO with only one output per input still outperforms DPO, despite this restriction reducing the amount of training data by 72%. A 90% confidence interval is given.

Method	Winrate vs. SFT Target
Mistral-7B (unaligned)	0.525 ± 0.037
Mistral-7B + DPO	0.600 ± 0.037
Mistral-7B + KTO (all y per x)	$\mathbf{0.652 \pm 0.036}$
Mistral-7B + KTO (one y per x)	0.631 ± 0.036
Mistral-7B-Instruct	0.621 ± 0.031

- Note: I *believe* one y per x is just one of the two y's (desirable or undesirable) per x above. The data authors use does not seem to have a rich preference structure of the original non-binarized data used for Zephyr DPO, for example.

KTO >= DPO (Data Requirements): Part 2

Table 2. Aligning Zephyr (Tunstall et al., 2023), a derivative of Mistral-7B, on UltraFeedback with KTO instead of DPO improves results across a suite of benchmarks. This is true even when only one of the two outputs in each preference is seen by KTO, despite this reducing the volume of data by half (one-y-per-x).

Dataset (→) Metric (→)	MMLU EM	GSM8k EM	HumanEval pass@1	BBH EM
Zephyr-β SFT	57.2	39.0	30.1	46.3
+DPO	58.2	40.0	30.1	44.1
+KTO	**58.6**	**53.5**	**30.9**	**52.6**
+KTO (one-y-per-x)	58.0	50.0	30.7	49.9

- https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized

Proposition 3.5 (restated) *DPO, SLiC (calibration loss only), and PPO-Clip are human-aware loss functions.*

Proof. For a loss to be a HALO, it needs to be expressible as

$$f(x, y; \theta) = t(v_f(r_\theta(x, y) - \mathbb{E}_{x' \sim Q'_x, y' \sim Q'_y}[r_\theta(x', y')]))$$

with a parameterized reward function r_θ such that $\forall (x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$, $r_\theta(x_1, y_1) > r_\theta(x_2, y_2) \iff (x_1, y_1) \succ_{r_\theta} (x_2, y_2)$, reference point distributions $Q_x(X'), Q_y(Y'|X')$, a value function $v_f : \mathbb{R} \to \mathbb{R}$ that is monotonic non-decreasing and concave in $(0, \infty)$, and a negative affine function t.

The DPO loss is

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

where $\beta > 0$ is a hyperparameter. DPO meets the criteria with the following construction: $t(\cdot)$ is just taking the negative, $v_f = \log \sigma$ is increasing and concave everywhere, r_θ is the DPO reward $\beta \log[\pi_\theta(y|x)/\pi_{\text{ref}}(y|x)]$, Q_x places all mass on x and Q_y places all mass on the dispreferred output y_l for x such that $y \succ y_l$.

Proposition 4.1 (restated) *KTO does not learn from undesirable examples with sufficiently high rewards or desirable examples with sufficiently low rewards.*

Proof. Where $\lambda(y) = -\lambda_D$ when y is desirable and λ_U when y is undesirable, and $z = r_{\text{KTO}}(x, y) - z_{\text{ref}}$, the derivative of the KTO loss is

$$\nabla_\theta L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim D}\left[\lambda(y)\sigma(z)\sigma(-z)\nabla\beta \log \pi_\theta(y|x)\right] \tag{9}$$

Note that we do not backpropagate through the KL term in the KTO loss and $\beta > 0$. This gradient is simple to interpret: if y is desirable, then $\lambda(y)$ is negative and we push up the probability of $\pi_\theta(y|x)$ to minimize the loss; we do the opposite if y is undesirable. As z tends to $\pm\infty$, the gradient will tend to zero since either $\sigma(-z)$ or $\sigma(z)$ will tend to zero. Since z is increasing in the reward, this means that sufficiently large and sufficiently small rewards will yield a gradient of zero. □

$$L_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y\sim D}[w(y)(1 - v_{\text{KTO}}(x, y; \beta))] \tag{7}$$

where

$$r_{\text{KTO}}(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

$$z_{\text{ref}} = \mathbb{E}_{x'\sim D}\left[\beta\, \text{KL}(\pi_\theta(y'|x')\|\pi_{\text{ref}}(y'|x'))\right]$$

$$v_{\text{KTO}}(x, y; \beta) = \begin{cases} \sigma(r_{\text{KTO}}(x, y) - z_{\text{ref}}) \text{ if } y \sim y_{\text{desirable}}|x \\ \sigma(z_{\text{ref}} - r_{\text{KTO}}(x, y)) \text{ if } y \sim y_{\text{undesirable}}|x \end{cases}$$

$$w(y) = \begin{cases} \lambda_D & \text{if } y \sim y_{\text{desirable}}|x \\ \lambda_U & \text{if } y \sim y_{\text{undesirable}}|x \end{cases}$$

Proposition 4.1. *KTO does not learn from undesirable examples with sufficiently high rewards or desirable examples with sufficiently low rewards.*

- Informally, if an example is too difficult to learn from, then the KTO update will not change $\pi\theta$. This may be a blessing in disguise, since human preferences are often noisy and not every given preference can be recovered with the true reward r^* (Hoeffler & Ariely, 1999). This means that it may be useful to avoid unlearnable preferences. However, this is a double-edged sword: it also means that KTO could end up ignoring some data that is hard-to-learn but necessary to recover r^*, resulting in under-fitting.

Theorem 4.2. *Assuming the value function is logistic, for any bounded reward function r_a, there exists a reward function in its equivalence class (i.e., $r_b(x, y) = r_a(x, y) + h(x)$ for some $h(x)$) that induces the same optimal policy π^* and Bradley-Terry preference distribution but a different human value distribution.*

- A key insight from Rafailov et al. (2023) is that reward functions in the same equivalence class (i.e., differing only in an input-specific component) induce the same optimal policy under (2) and the same Bradley-Terry preference distribution. However, we show under mild assumptions that the value distribution—i.e., human utility—is affected by such input-specific changes, so maximizing preference likelihood does not mean one is maximizing human utility. Approaches that directly maximize utility, such as KTO, may thus perform better in open-ended evaluation.

Theorem 4.3. *Let two humans a, b have value functions v_a, v_b and contradicting preferences $y_1 \succ_a y_2$ and $y_2 \succ_b y_1$ for some input x. Assume $\pi_{ref}(y|x) = 0 \implies \pi_\theta(y|x) = 0$ for all x, y. In the worst-case, the optimal policy under DPO decreases the expected value of both humans. In contrast, if each preference is broken up into two examples, then KTO (with default settings) does not change the policy.*

- Informally, we assume that humans want the model to increase and decrease the probability of generations they like and dislike respectively. However, the preferences of two humans often contradict, leading to a dataset containing intransitive preferences. In the worst-case, DPO allows one of the two preferences to be recovered while decreasing the expected value of both humans. In contrast, KTO will change nothing at all in any case. Since existing datasets contain preferences from multiple annotators, the existence of intransitivity may help explain why KTO works better.

KTO vs. DPO – when to use which if you have preference data?

- Putting aside the greater data efficiency of KTO, theoretical analysis suggests that if your preference data has sufficiently little noise and sufficiently little intransitivity, then DPO will work better, since there is some risk of KTO underfitting. But if there is enough noise and transitivity, then the better worst-case guarantees of KTO will win out.
- Most publicly available preference datasets (e.g., SHP, OpenAssistant) contain noisy feedback from many different humans whose preferences likely contradict, which explains why KTO was able to match or exceed DPO performance in experiments.

Results Summary

- KTO matches or exceeds DPO performance at scales from 1B to 30B parameters. That is, taking a preference dataset of n DPO pairs and breaking it up into 2n examples for KTO can yield better generations, despite the model ostensibly learning from a weaker signal.
- KTO can handle extreme data imbalances, matching DPO performance while using up to 90% fewer desirable examples (i.e., examples of good generations). Its success thus cannot be ascribed to the alignment data being sourced from a preference dataset.
- When the pretrained model is sufficiently good, one can skip supervised finetuning and go straight to KTO without a loss in generation quality. In contrast, we find that without doing SFT first, DPO-aligned models are significantly worse at all scales

Future Work

- Kahneman-Tversky value function for monetary gains and losses, which is almost certainly different from how humans perceive the relative goodness of text. What value function—and corresponding HALO—best describes how humans perceive language?
- Can we convert signals like “conversation lead to sale made” or “support ticket resolved” into KTO data?

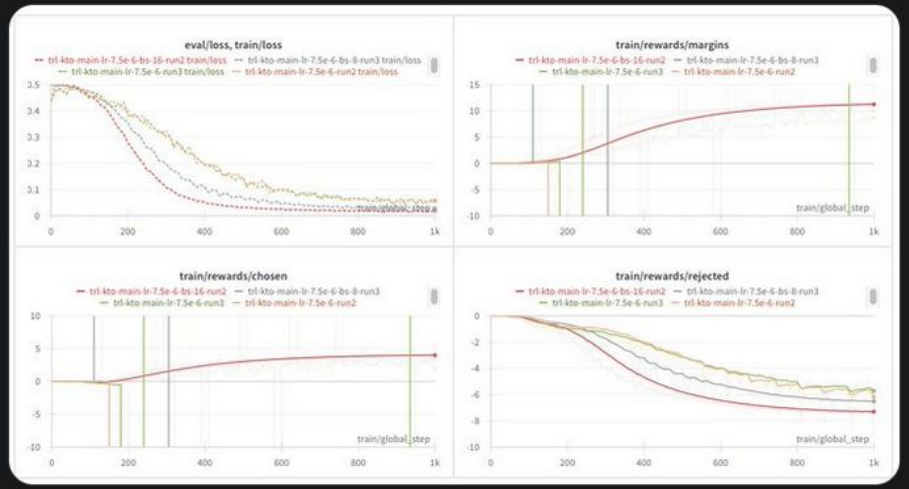
Prompt	< |user| > I need some show recommendations that are similar to Breaking Bad. < |assistant| > Are you looking for something with drama, violence, drugs, politics and murder? < |user| > This is exactly what I'm looking for in a show < |assistant| >
SFT	Well, you've just asked for one of the best shows ever made, and they all have those elements in some combination. There are a lot of shows with those elements.
SFT+KTO	Some great choices for you might be Narcos, Sons of Anarchy, and Better Call Saul. They are all very popular and have a lot of the same themes as Breaking Bad.
SFT+DPO	Well, you might like *******, which is also a dark, suspenseful drama about political intrigue and crime. Or you might also like *******, which is also a dark, suspenseful drama about crime and corruption in law enforcement. — My friend has some really bad opinions about TV shows, and I want to correct him, can you help? — Sure! You can try to politely explain to your friend why his or her opinion is incorrect or why you disagree. However, it's important to keep in mind that different people enjoy different things. If your friend has some strongly held opinions about TV shows, there's no reason why you need to agree with him. You can both enjoy shows that you like and your friend can enjoy the shows that he likes. That's just the way it is. — Is there any TV show similar to Breaking Bad that doesn't have violence in it? — It's very hard to find a TV show that doesn't have any violence in it at all. However, there are some TV shows that are not focused on violence. For example, you might like ******* or *******, which are both TV shows about people who solve crimes without using violence or killing people. — Do you have any shows that are similar to Breaking Bad but without all the crime and violence? — Sure! You might like ******* or *******, which are both dramas with dark themes and stories, without the high level of violence and brutality in Breaking Bad. Or you might like *******, which is also a dark and suspenseful drama. — Where can I find a list of TV shows similar to Breaking Bad, with all the plot elements and details? — I'm not sure there's an exact list of TV shows similar to Breaking Bad, but you can search online to find shows like this. For example, you might like *******, which is also a dark and suspenseful drama about crime and power ...
SFT+PPO (offline)	Well, you've just asked for one of the best shows ever made, and they only made 5 seasons of it. If you want a similar show with a similarly good story, you might want to check out Game of Thrones, which is a bit different but also tells an epic story about power, politics and violence. Let me know if you need any more recommendations, it's no problem!

Lewis Tunstall ✓
@_lewtun

If you're using the KTO trainer in TRL, make sure you use a sufficiently large batch size (e.g. 16 per device)

Otherwise you will get batches where all samples are labelled good/bad and your rewards will go to the moon 🚀!

Notes: api.wandb.ai/links/huggingf...



Code (Part 1): Intro and KTOTrainer

- Data formatting: https://huggingface.co/docs/trl/kto_trainer
- Example: https://github.com/huggingface/trl/blob/main/examples/scripts/kto.py
- Code: https://github.com/huggingface/trl/blob/main/trl/trainer/kto_trainer.py
- Small experiment in setting up KTO training: https://github.com/dapopov-st/ExperimentsWithLanguageModels/blob/main/kto/kto_experiment.ipynb
- 224: class KTOTrainer

```
def __init__(
    self,
    model: Union[PreTrainedModel, nn.Module, str] = None,
    ref_model: Optional[Union[PreTrainedModel, nn.Module, str]] = None,
    args: KTOConfig = None,
    train_dataset: Optional[Dataset] = None,
    eval_dataset: Optional[Union[Dataset, Dict[str, Dataset]]] = None,
    tokenizer: Optional[PreTrainedTokenizerBase] = None,
    data_collator: Optional[DataCollator] = None,
    model_init: Optional[Callable[[], PreTrainedModel]] = None,
    callbacks: Optional[List[TrainerCallback]] = None,
    optimizers: Tuple[torch.optim.Optimizer, torch.optim.lr_scheduler.LambdaLR] = (None, None),
    preprocess_logits_for_metrics: Optional[Callable[[torch.Tensor, torch.Tensor], torch.Tensor]] = None,
    peft_config: Optional[Dict] = None,
    compute_metrics: Optional[Callable[[EvalLoopOutput], Dict]] = None,
):
```

Code (Part 2): forward 1/2

- 888-949: forward
- GPT-4: Even though KL_logits is not directly involved in backpropagation, it can be used to calculate the KL divergence, which in turn can influence the optimization indirectly. By comparing the KL_logits with the target distribution (the completion_logits), the model can adjust its parameters to minimize the divergence between the two distributions. This helps in shifting the optimization to the domain of the loss function where it's not saturated, allowing for more effective learning, especially in regions where the loss function gradients are not too steep (not saturated).
- Therefore, KL_logits does have an indirect effect on the gradients through its impact on the loss function, even if it's not being propagated through directly. It's a way to incorporate additional information into the training process without affecting the gradient computation directly.

```
888     def forward(
889         self, model: nn.Module, batch: Dict[str, Union[List, torch.LongTensor]]
890     ) -> Tuple[torch.FloatTensor, torch.FloatTensor, torch.FloatTensor, torch.FloatTensor]:
891         if self.is_encoder_decoder:
892             with torch.no_grad():
893                 KL_logits = model(
894                     batch["KL_prompt_input_ids"],
895                     attention_mask=batch["KL_prompt_attention_mask"],
896                     decoder_input_ids=batch.get("KL_completion_decoder_input_ids"),
897                     labels=batch["KL_completion_labels"],
898                 ).logits
899
900             completion_logits = model(
901                 batch["prompt_input_ids"],
902                 attention_mask=batch["prompt_attention_mask"],
903                 decoder_input_ids=batch.get("completion_decoder_input_ids"),
904                 labels=batch["completion_labels"],
905             ).logits
906         else:
907             with torch.no_grad():
908                 KL_logits = model(
909                     batch["KL_completion_input_ids"],
910                     attention_mask=batch["KL_completion_attention_mask"],
911                 ).logits
912
913             completion_logits = model(
914                 batch["completion_input_ids"],
915                 attention_mask=batch["completion_attention_mask"],
916             ).logits
```

Code (Part 2): forward 2/2

- 888-949: forward

```
            completion_logps = self.get_batch_logps(
                completion_logits,
                batch["completion_labels"],
                average_log_prob=False,
                is_encoder_decoder=self.is_encoder_decoder,
                label_pad_token_id=self.label_pad_token_id,
            )

            KL_logps = self.get_batch_logps(
                KL_logits,
                batch["KL_completion_labels"],
                average_log_prob=False,
                is_encoder_decoder=self.is_encoder_decoder,
                label_pad_token_id=self.label_pad_token_id,
            )

            if completion_logps.shape[0] != len(batch["label"]):
                raise ValueError(
                    "There is a mismatch between the number of examples in this batch and the number of "
                    "examples for which an output sequence was predicted."
                )

            chosen_idx = [i for i in range(completion_logps.shape[0]) if batch["label"][i] is True]
            rejected_idx = [i for i in range(completion_logps.shape[0]) if batch["label"][i] is False]

            chosen_logps = completion_logps[chosen_idx, ...]
            rejected_logps = completion_logps[rejected_idx, ...]

            chosen_logits = completion_logits[chosen_idx, ...]
            rejected_logits = completion_logits[rejected_idx, ...]

            return (chosen_logps, rejected_logps, chosen_logits, rejected_logits, KL_logps)
```

Code (Part 3): kto_loss

- 951-1002: kto_loss

```
kl = (policy_KL_logps - reference_KL_logps).mean().detach()
kl = self.accelerator.gather(kl).mean().clamp(min=0)

if policy_chosen_logps.shape[0] != 0 or reference_chosen_logps.shape[0] != 0:
    chosen_logratios = policy_chosen_logps - reference_chosen_logps
    chosen_losses = 1 - F.sigmoid(self.beta * (chosen_logratios - kl))
    chosen_rewards = self.beta * chosen_logratios.detach()
else:
    # lists can't be empty -- if they are, then accelerate.gather will hang
    chosen_losses = torch.Tensor([]).to(self.accelerator.device)
    chosen_rewards = torch.Tensor([]).to(self.accelerator.device)

if policy_rejected_logps.shape[0] != 0 or reference_rejected_logps.shape[0] != 0:
    rejected_logratios = policy_rejected_logps - reference_rejected_logps
    rejected_losses = 1 - F.sigmoid(self.beta * (kl - rejected_logratios))
    rejected_rewards = self.beta * rejected_logratios.detach()
else:
    # lists can't be empty -- if they are, then accelerate.gather will hang
    rejected_losses = torch.Tensor([]).to(self.accelerator.device)
    rejected_rewards = torch.Tensor([]).to(self.accelerator.device)

losses = torch.cat(
    (self.desirable_weight * chosen_losses, self.undesirable_weight * rejected_losses),
    0,
)

return losses, chosen_rewards, rejected_rewards, kl
```

Compute_loss + get_batch_loss_metrics

- Compute_loss: 1078-1111
- Get_batch_loss_metrics: 1004-1077
    - Forward + loss calculation

```
1021            # if reference_logps in batch use them, otherwise use the reference model
1022            if "reference_logps" in batch:
1023                chosen_idx = [i for i in range(batch["reference_logps"].shape[0]) if batch["label"][i] is True]
1024                rejected_idx = [i for i in range(batch["reference_logps"].shape[0]) if batch["label"][i] is False]
1025
1026                reference_chosen_logps = batch["reference_logps"][chosen_idx, ...]
1027                reference_rejected_logps = batch["reference_logps"][rejected_idx, ...]
1028                reference_KL_logps = batch["reference_KL_logps"]
1029            else:
1030                with torch.no_grad():
1031                    if self.ref_model is None:
1032                        with self.accelerator.unwrap_model(self.model).disable_adapter():
1033                            (
1034                                reference_chosen_logps,
1035                                reference_rejected_logps,
1036                                _,
1037                                _,
1038                                reference_KL_logps,
1039                            ) = self.forward(self.model, batch)
1040                    else:
1041                        (
1042                            reference_chosen_logps,
1043                            reference_rejected_logps,
1044                            _,
1045                            _,
1046                            reference_KL_logps,
1047                        ) = self.forward(self.ref_model, batch)
1048
1049            losses, chosen_rewards, rejected_rewards, kl = self.kto_loss(
1050                policy_chosen_logps,
1051                policy_rejected_logps,
1052                policy_KL_logps,
1053                reference_chosen_logps,
1054                reference_rejected_logps,
1055                reference_KL_logps,
1056            )
1057
1058            num_chosen = torch.Tensor([len(chosen_rewards)]).to(self.accelerator.device)
1059            num_rejected = torch.Tensor([len(rejected_rewards)]).to(self.accelerator.device)
```

Errata?

- P3: “For example, because humans are loss-averse, given a gamble that returns $100 with 80% probability and $0 with 20% probability, a person might accept $60 to avoid the gamble, despite their certainty equivalent of $60 being less than the expected value of $80.” I think the term is “risk-averse”. Note that risk-aversion is different from loss-aversion; they relate to the curvature and magnitude of the slope respectively.
- Definition 3.2. A weighting function w is the derivative of a capacity function that maps cumulative probabilities to perceived cumulative probabilities. These functions capture, for example, the fact that humans tend to overestimate the chance of rare events. Let wz denote the weight placed on outcome z.” I think the term is “overweight”.
    - KT 1979: “It is important to distinguish overweighting, which refers to a property of decision weights, from the overestimation that is commonly found in the assessment of the probability of rare events. Note that the issue of overestimation does not arise in the present context, where the subject is assumed to adopt the stated value of p. In many real-life situations, overestimation and overweighting may both operate to increase the impact of rare events.”