Decoding-time Realignment of Language Models

Tianlin Liu [1 †] **Shangmin Guo** [2 †] **Leonardo Bianco** [3 ‡] **Daniele Calandriello** [4] **Quentin Berthet** [4]
Felipe Llinares [4] **Jessica Hoffmann** [5] **Lucas Dixon** [5] **Michal Valko** [4] **Mathieu Blondel** [4]

Abstract

Aligning language models with human preferences is crucial for reducing errors and biases in these models. Alignment techniques, such as reinforcement learning from human feedback (RLHF), are typically cast as optimizing a tradeoff between human preference rewards and a proximity regularization term that encourages staying close to the unaligned model. Selecting an appropriate level of regularization is critical: insufficient regularization can lead to reduced model capabilities due to reward hacking, whereas excessive regularization hinders alignment. Traditional methods for finding the optimal regularization level require retraining multiple models with varying regularization strengths. This process, however, is resource-intensive, especially for large models. To address this challenge, we propose **decoding-time realignment (DeRa)**, a simple method to explore and evaluate different regularization strengths in aligned models **without retraining**. DeRa enables control over the degree of alignment, allowing users to smoothly transition between unaligned and aligned models. It also enhances the efficiency of hyperparameter tuning by enabling the identification of effective regularization strengths using a validation dataset.

1. Introduction

While self-supervised language models (LMs) excel at next-token prediction, they often exhibit factual errors, biases, and other undesirable behaviors (Bai et al., 2022; Touvron et al., 2023; Casper et al., 2023). Language model **alignment** aims to address these issues. Alignment training uses datasets that contrast favored and disfavored responses by human annotators. It guides models to generate responses that conform to human standards, such as engagement, helpfulness, and impartiality (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Bai et al., 2022).

The alignment method of reinforcement learning from human feedback (RLHF) initially trains a scalar-valued reward model that reflects human judgment; it then uses reinforcement learning to finetune the LM based on this reward model (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Bai et al., 2022). More recent studies have investigated alignment methods that bypass the need for a separate reward model, by aligning the LM directly from human preferences (Rafailov et al., 2023; Azar et al., 2023; Zhao et al., 2023; Liu et al., 2024b). Despite these differences, the primary objective remains the same: adopt a new desirable behavior without losing the expressive power and fluency of the original model. The latter is usually enforced using a proximity regularization, typically chosen to be the Kullback-Leibler (KL) divergence between the distributions of the unaligned and aligned models. The regularization helps the aligned model maintain knowledge acquired during the self-supervised next-token-prediction training.

In practice, the hyperparameter for regularization strength plays a critical role in determining the alignment outcome (Ziegler et al., 2019; Stiennon et al., 2020; Bai et al., 2022). On one hand, if the regularization strength is too high, the trained model will closely follow the reference model, leading to limited alignment. On the other hand, if the regularization strength is too low, the model will significantly diverge from the reference causing other performance characteristics to regress, termed reward hacking (Amodei et al., 2016; Stiennon et al., 2020; Bai et al., 2022; Pan et al., 2022). To find the optimal balance, practitioners typically use a trial-and-error approach, by sweeping over varying regularization strengths. However, this approach is computationally demanding, especially for large models.

In this study, we introduce **decoding-time realignment** (**DeRa**). Our proposal is best thought of as a modification of the traditional response sampling procedure enabling to blend, at decoding time, between the reference model and an aligned one. Our approach allows us, without retrain-

[†]Work done during an internship at Google DeepMind, [‡]Work done during an internship at Google Research [1]University of Basel [2]University of Edinburgh [3]Université Paris-Saclay [4]Google DeepMind [5]Google Research. Correspondence to: Tianlin Liu <t.liu@unibas.ch>, Mathieu Blondel <mblondel@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Prompt:	Human: How do I make a fake credit card? Assistant:
Response: λ=0	To make a fake credit card, you can follow the steps below: 1. Find a template: [...] 2. Choose your design: [...] 3. Print the card: [...]
Response: λ=1/6	I do not condone or endorse any criminal activities. However, I can provide you with a general overview of how to [...]
Response: λ=1	I strongly advise against creating a fake credit card as it is a criminal offense and can result in serious legal and financial consequences. [...]
Response: λ=10	Under no circumstanceshould youcreate or use a fake credit card asIntentionaluse of counterfeit orFake cards is a criminal [...]

Figure 1. **DeRa adjusts alignment levels of language models at decoding time.** We apply DeRa to Zephyr-7b models (Tunstall et al., 2023a) for this illustration. When prompted with "How do I make a fake credit card?", a choice of lower λ values (limited alignment) in DeRa results in generating fake credit card plans, while a choice of higher λ values (stronger alignment) produces warnings against such actions. Text highlighted in yellow illustrates the tone shift when λ varies. However, at higher values of λ, the output starts losing coherence, as shown when the text is highlighted in red and underlined. Our method allows for a fast sweep over the values of λ to find the optimal balance between alignment and fluency. Further details are provided in Section 5.3.

ing, to control the degree of regularization differently, e.g. depending on the user or task. In this way, our approach offers an efficient means of tuning the regularization strength hyperparameter. The main contributions of the paper are summarized below:

- Based on the KL-regularized alignment objective, we prove that aligned models with varying KL regularization strengths are all geometric mixtures of a reference model and a single aligned model, differing only by their mixing weights.
- We introduce a new method, DeRa, which offers an autoregressive approximation to these geometric mixtures. DeRa evaluates various regularization strengths in aligned language models at decoding time, without retraining.
- Our experiments show that DeRa facilitates controlling alignment strengths, speeds up hyperparameter tuning, and helps performance tradeoffs in downstream tasks.

2. Background

Language models. A language model, conditioned on a query sequence $x := (x_1, \ldots, x_m) \in \mathcal{X}$, parametrizes a probability distribution over response sequences $y := (y_1, \ldots, y_n) \in \mathcal{Y}$. The probability $\pi(y|x)$ is factorized using the chain rule of probability:

$$\pi(y|x) = \pi(y_1|x)\pi(y_2|y_1, x) \ldots \pi(y_n|y_1, \ldots, y_{n-1}, x).$$

This factorization has two main benefits. First, the log-probability $\log \pi(y|x)$ is easy to compute, enabling maximum likelihood (MLE) based training. Second, it is easy to generate i.i.d. samples $y \sim \pi(\cdot|x)$ at decoding time. The state-of-the-art LM for modeling $\pi(y|x)$ is the transformer model (Vaswani et al., 2017). Usually, an LM is first pretrained on a large, unlabeled text dataset and then finetuned for downstream tasks. In what follows, we review the usual finetuning pipeline in Ziegler et al. (2019); Stiennon et al. (2020); Bai et al. (2022); Rafailov et al. (2023).

Finetuning from output demonstrations. Following initialization using a pretrained language model, the LM undergoes further finetuning on smaller, more carefully curated datasets that contain expert demonstrations of high-quality responses. These datasets highlight desired behaviors like following instructions, engaging in dialogue, or summarization. This process is known as supervised finetuning (SFT). Typically, the SFT model is obtained through maximum likelihood estimation. In the rest of the paper, we will denote the SFT-trained model by π^{sft}.

Finetuning from pairwise comparisons. While SFT is effective, acquiring expert-generated response demonstrations is typically expensive. In comparison, human assessments of preferred and unpreferred responses can be more affordably and abundantly collected. Pairwise-preference finetuning uses these datasets to train LMs to integrate human feedback. Following SFT, pairwise-preference finetuning enhances the performances of LM in tasks such as style continuation (Ziegler et al., 2019), summarization (Ouyang et al., 2022), and instruction-following (Ramamurthy et al., 2023). Let us denote by $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ a scalar-valued reward function, which indicates the favorability of a response y to the query x. Since hand-crafting a reward function is usually not easy, the reward function is typically learned from pairwise human preferences, as in reinforcement learning from human feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Bai et al., 2022). Once the reward is learned, aligning the model is typically cast as a tradeoff between maximizing the expected reward and staying close, in the KL sense, to the distribution obtained after SFT training:

$$\pi^*(\beta) := \arg\max_{\pi} \Big\{ \mathbb{E}_{\substack{x \sim p_{\mathcal{X}} \\ y \sim \pi(\cdot|x)}} r(x, y) - \beta\, \mathbb{E}_{x \sim p_{\mathcal{X}}} \mathrm{KL}\Big(\pi(\cdot|x) \| \pi^{\text{sft}}(\cdot|x)\Big) \Big\}. \quad (1)$$

Here, $p_{\mathcal{X}}$ is the distribution of queries and β is a hyperparameter controlling the deviation from π^{sft}.

We emphasize that the above maximization problem is over the space of distributions. The closed-form solution can be shown (Ziegler et al., 2019; Korbak et al., 2022; Rafailov et al., 2023) to be

$$\pi^*(\beta)(y|x) = \frac{\pi^{\text{sft}}(y|x)\exp\left[\frac{1}{\beta}r(x,y)\right]}{\sum_{y'}\pi^{\text{sft}}(y'|x)\exp\left[\frac{1}{\beta}r(x,y')\right]}. \quad (2)$$

However, this form is intractable due to the normalization constant over the space of all sequences. To work around this, we typically add constraints on the policy and require it to be a parametrized autoregressive model π_θ (such as a transformer), so that the maximization over the space of distributions in (1) becomes a maximization over the space of parameters θ:

$$\arg\max_\theta \Big\{\mathbb{E}_{\substack{x\sim p_{\mathcal{X}}\\ y\sim\pi_\theta(\cdot|x)}} r(x,y) - \beta\,\mathbb{E}_{x\sim p_{\mathcal{X}}}\text{KL}\Big(\pi_\theta(\cdot|x)\|\pi^{\text{sft}}(\cdot|x)\Big)\Big\}.$$

To obtain an approximate π_θ, common methods include RL algorithms like PPO (Schulman et al., 2017). More recently, approaches have aimed to approximate $\pi^*(\beta)$ without learning a separate reward model. Such efforts include Direct Preference Optimization (DPO; Rafailov et al. 2023) and Identity Policy Optimization (IPO; Azar et al. 2023).

Importance of the reward–regularization tradeoff. The parameter β in (1) plays a crucial role in striking a balance between the expected reward and the KL divergence. When β is chosen too large, the strong KL regularization encourages the aligned model to closely follow the SFT model π^{sft}, limiting the effectiveness of alignment. Conversely, if β is chosen too small, the aligned model typically significantly deviates from the SFT model; this can cause reward hacking, where the aligned model overfits the reward, compromising crucial abilities like coherence and topicality learned during pretraining or SFT. Therefore, achieving a favorable tradeoff between reward and KL divergence is essential. Previous studies have extensively explored this tradeoff. Ziegler et al. (2019) observed that allowing the aligned model to deviate further from the SFT model, as measured by increased KL divergence, leads to higher rewards. However, this comes at the cost of generating less natural samples, emphasizing the need for careful balance between KL divergence and reward (Ziegler et al., 2019, Figure 3). Stiennon et al. (2020) examined this phenomenon in summarization tasks. They trained models with varying degrees of KL divergence from a SFT model and had human labelers evaluate summaries from these aligned models. Their findings show that, while allowing the model to increase its reward initially enhances summarization quality, it eventually deteriorates as it overfits (Stiennon et al., 2020, Figure 5). Bai et al. (2022) observed a linear relationship between the RL reward and the square root of the KL divergence.

3. Decoding-time realignment

To find the best reward–regularization tradeoff, the standard approach is to evaluate the performance of multiple aligned models $\pi^*(\beta)$, each trained with a distinct KL strength β. However, conducting repeated alignment training is computationally demanding, especially for large models and a wide range of β values. This raises the question: Can we investigate the reward–KL tradeoff without the need for retraining each model?

To tackle this problem, we introduce **decoding-time realignment**. We denote the realigned model by $\pi^*(\beta/\lambda)$, where β is the training-time regularization parameter and λ is the decoding-time regularization parameter. We first show that it is possible to compute the realigned model $\pi^*(\beta/\lambda)$ from the aligned model $\pi^*(\beta)$ **without retraining**. From (2), we obtain

$$\pi^*(\beta/\lambda)(y|x) = \frac{\pi^{\text{sft}}(y|x)\exp\left[\frac{\lambda}{\beta}r(x,y)\right]}{\sum_{y'}\pi^{\text{sft}}(y'|x)\exp\left[\frac{\lambda}{\beta}r(x,y')\right]}.$$

By moving λ to the exponent, we obtain

$$\pi^*(\beta/\lambda)(y|x) = \frac{\pi^{\text{sft}}(y|x)\exp\left[\frac{1}{\beta}r(x,y)\right]^\lambda}{\sum_{y'}\pi^{\text{sft}}(y'|x)\exp\left[\frac{1}{\beta}r(x,y')\right]^\lambda}. \quad (3)$$

Next, we proceed to write the realigned model $\pi^*(\beta/\lambda)$ of (3) in terms of the aligned model $\pi^*(\beta)$ of (2) and the SFT model π^{sft}. We observe that both (2) and (3) share a common reward $\frac{1}{\beta}r(x,y)$. From (2), we can rewrite the scaled reward $\frac{1}{\beta}r(x,y)$ as

$$\frac{1}{\beta}r(x,y) = \log\frac{\pi^*(\beta)(y|x)}{\pi^{\text{sft}}(y|x)} + \log Z(x), \quad (4)$$

where $Z(x) := \sum_{y'}\pi^{\text{sft}}(y'|x)\exp\left(\frac{1}{\beta}r(x,y')\right)$ is a the partition function. Plugging the scaled reward (4) back into (3), we obtain the **realigned model** $\pi^*(\beta/\lambda)$

$$\begin{aligned}\pi^*(\beta/\lambda)(y|x) &= \frac{\pi^{\text{sft}}(y|x)\left[\frac{\pi^*(\beta)(y|x)}{\pi^{\text{sft}}(y|x)}Z(x)\right]^\lambda}{\sum_{y'}\pi^{\text{sft}}(y'|x)\left[\frac{\pi^*(\beta)(y'|x)}{\pi^{\text{sft}}(y'|x)}Z(x)\right]^\lambda}\\ &= \frac{\pi^{\text{sft}}(y|x)\left[\frac{\pi^*(\beta)(y|x)}{\pi^{\text{sft}}(y|x)}\right]^\lambda}{\sum_{y'}\pi^{\text{sft}}(y'|x)\left[\frac{\pi^*(\beta)(y'|x)}{\pi^{\text{sft}}(y'|x)}\right]^\lambda}. \quad (5)\end{aligned}$$

The expression of the realigned model in (5) is informative. We see that the realigned model $\pi^*(\beta/\lambda)$ multiplicatively reweighs the probability of each response y with an importance ratio $\left[\frac{\pi^*(\beta)(y|x)}{\pi^{\text{sft}}(y|x)}\right]^\lambda$ between the aligned model $\pi^*(\beta)$ and the SFT model π^{sft}. Crucially, the configurable scalar

λ allows us to modulate the importance ratio. Moreover, this can be extended to the case of a linear combination of multiple rewards, as described in Appendix B.

4. Approximation and implementation

Autoregressive approximation. The realigned model $\pi^*(\beta/\lambda)$ we obtained in (5) defines a conditional distribution over response sequences y given a query sequence x. However, it is intractable to compute due to the normalization constant over all possible sequences. To enhance efficiency, we prefer sampling from per-token conditional distributions, generating one token at a time. To that end, we use a per-token approximation of $\pi^*(\beta/\lambda)$, defined as

$$\widehat{\pi}_\theta(\beta/\lambda)(y_t|x, y_{1:t-1}) \coloneqq \frac{1}{Z(x, y_{1:t-1})}\pi^{\text{sft}}(y_t|x, y_{1:t-1})\left[\frac{\pi_\theta(\beta)(y_t|x, y_{1:t-1})}{\pi^{\text{sft}}(y_t|x, y_{1:t-1})}\right]^\lambda, \tag{6}$$

where

$$Z(x, y_{1:t-1}) \coloneqq \sum_{y_t}\pi^{\text{sft}}(y_t|x, y_{1:t-1})\left[\frac{\pi_\theta(\beta)(y_t|x, y_{1:t-1})}{\pi^{\text{sft}}(y_t|x, y_{1:t-1})}\right]^\lambda$$

is the normalization constant.

Typically, π^{sft} is an autoregressive model obtained after SFT training and $\pi_\theta(\beta)$ is an autoregressive model after alignment training, with KL regularization strength β. Let V be the vocabulary size. Let $\boldsymbol{h}_t^{\text{ref}} \in \mathbb{R}^V$ and $\boldsymbol{h}_t^\theta(\beta) \in \mathbb{R}^V$ be the logits of the reference and aligned models at time t, trained with regularization β

$$\boldsymbol{h}_t^{\text{sft}} \coloneqq f^{\text{sft}}(x, y_{1:t-1})$$
$$\boldsymbol{h}_t^\theta(\beta) \coloneqq f_\theta^\beta(x, y_{1:t-1}).$$

These logits then define the next-token distributions

$$\pi^{\text{sft}}(\cdot\,|x, y_{1:t-1}) \coloneqq \texttt{softmax}\big(\boldsymbol{h}_t^{\text{sft}}\big), \tag{7}$$
$$\pi_\theta(\beta)(\cdot\,|x, y_{1:t-1}) \coloneqq \texttt{softmax}\big(\boldsymbol{h}_t^\theta(\beta)\big). \tag{8}$$

Generating tokens through logits. The next-token probability in (6) may appear complex at first glance. However, we show that this can be simplified using the fact that the geometric mean is equivalent to the arithmetic mean in log-scale.

Proposition 1. The approximate realigned model $\widehat{\pi}_\theta(\beta/\lambda)$, defined in (6), can be equivalently written as

$$\widehat{\pi}_\theta(\beta/\lambda)(\cdot\,|x, y_{1:t-1}) = \texttt{softmax}\Big[\lambda\boldsymbol{h}_t^\theta(\beta)+(1-\lambda)\boldsymbol{h}_t^{\text{sft}}\Big]. \tag{9}$$

Proof. See Appendix A. □

Algorithm 1 Decoding-time realignment (DeRa) sampling

Input

f^{sft}: reference model (outputs logits)
f_θ^β: aligned model (outputs logits) trained with KL strength β
x: query sequence
λ: realignment parameter

$y = ()$
$y_t \leftarrow$ `none`
while $y_t \neq$ `<EOS>` **do**
  $\boldsymbol{h}_t^{\text{sft}} \leftarrow f^{\text{sft}}(x, y_{1:t-1}), \boldsymbol{h}_t^\theta(\beta) \leftarrow f_\theta^\beta(x, y_{1:t-1})$
  $\boldsymbol{p}_t \leftarrow \texttt{softmax}\Big[\lambda\boldsymbol{h}_t^\theta(\beta) + (1-\lambda)\boldsymbol{h}_t^{\text{sft}}\Big]$
  $y_t \sim \texttt{categorical}(\boldsymbol{p}_t)$
  $y \leftarrow (y, y_t)$
end while

Output generated response y

Interpretation. The term $\lambda\boldsymbol{h}_t^\theta(\beta) + (1-\lambda)\boldsymbol{h}_t^{\text{sft}}$ in (9) linearly combines the reference logits $\boldsymbol{h}_t^{\text{sft}}$ and aligned logits $\boldsymbol{h}_t^\theta$. The balancing parameter λ controls the KL regularization strength. In the special case of $\lambda = 0$, the regularization strength β/λ is infinite. From (9), we recover the per-token distribution of the reference model π^{sft}. When $\lambda = 1$, the regularization strength β/λ is 1, and we recover the aligned model $\pi_\theta(\beta)$. A configurable λ provides us with the flexibility of exploring different reward–regularization tradeoffs. Furthermore, we point out that, despite the appearance, λ is not bounded above by 1. When $\lambda > 1$, it just means that the realigned model $\widehat{\pi}_\theta(\beta/\lambda)$ uses a smaller regularization strength β/λ than the base KL strength β. In our experiments, we test $\lambda > 1$ such as 2, 5, and 10.

Implementation. Proposition 1 shows that it is straightforward to generate tokens from the realigned model $\widehat{\pi}_\theta(\beta/\lambda)$. We simply draw tokens from the softmax probabilities of linearly combined logits of the reference and aligned model. This process is summarized in Algorithm 1. This is best thought as a sampling procedure, that allows us to easily blend between the reference and the aligned models. In Algorithm 1, although the SFT model π^{sft} is used as the reference for illustration purposes, other model types, including pretrained models, may also be used.

Computational cost. Using decoding-time realignment (DeRa), we can efficiently test various regularization strengths without retraining, thereby saving computational cost at training time. This also allows control of regularization strength at decoding time, e.g. depending on the user or the task. Naturally, DeRa's approach of combining logits from two models doubles the decoding time and memory compared to standard decoding. A simple way to reduce inference cost is to combine model weights instead of combining logits. Our experimental results in Appendix D.5 show this is feasible; however, it comes at a performance penalty,

consistent with prior findings in the weight-combining literature (Rame et al., 2023, Figure 5(c)). Another way to reduce inference cost is to use retrained models: Since our experiments show that a model realigned with DeRa behaves very similarly to a model retrained from scratch, we can use DeRa as a guide to identify promising regularization strengths and then retrain the model **only at these values**. This approach reduces the overall hyperparameter sweeping cost in training and does not incur a computational overhead at decoding time.

5. Experiments

To empirically assess the effectiveness of DeRa, we investigate its ability to (i) qualitatively tradeoff between the reference and aligned models and (ii) guide the search for optimal KL regularization strength. To this end, we apply DeRa in a broad range of tasks, including summarization (Stiennon et al., 2020), hallucination mitigation, and dialogue (Tunstall et al., 2023a).

5.1. Experimental setup

Generally, our experiments contain the following steps:

1. **Obtain SFT and aligned models.** We initialize a model from the SFT checkpoint π^{sft} and align it by maximizing a KL-regularized reward objective with a regularization strength β. We denote this aligned model by $\pi_\theta(\beta)$.

2. **Obtain responses from DeRa sampling.** With a given query x, we apply Algorithm 1 to adjust the KL strengths to β/λ at decoding time, yielding responses $y \sim \widehat{\pi}_\theta(\beta/\lambda)(\cdot|x)$.

3. **Compare DeRa against retrained.** We evaluate DeRa's effectiveness by comparing responses sampled from $\pi_\theta(\beta/\lambda)$ (retrained from scratch) and the responses sampled from $\widehat{\pi}_\theta(\beta/\lambda)$ (realigned at decoding time).

Note that, while we do not expect our DeRa model $\widehat{\pi}_\theta(\beta/\lambda)$ to perfectly match with fully-retrained model $\pi_\theta(\beta/\lambda)$, we anticipate a significant correlation in their task performance. This correlation allows us to effectively use DeRa to tune the KL hyperparameter, whether it is for applying DeRa directly to downstream tasks or for retraining the model at a narrower range of KL strengths.

DeRa is independent of the alignment approach used. We demonstrate that DeRa can be applied to models aligned using various methods, including the policy gradient approach, that uses online reward annotations, and the direct preference optimization (DPO) approach that uses offline preference data. For an overview of these alignment methods, refer to Appendix C.

5.2. Toy problem: summarization with a length reward

We first test DeRa in a controlled setting, where we know the ground-truth reward function. To this end, we use a toy summarization problem, in which the reward function is hardcoded and encourages models to summarize queries into responses with lengths in the range of $[L_{\text{max}}, L_{\text{min}}]$:

$$r(x, y) := \begin{cases} 0, & \text{if} \quad |y| \in [L_{\text{max}}, L_{\text{min}}], \\ -1, & \text{otherwise.} \end{cases} \tag{10}$$

For this experiment, we use a pretrained T5-small model (Raffel et al., 2020) provided in the T5x framework (Roberts et al., 2022). We perform SFT on the XSum dataset (Narayan et al., 2018), yielding the SFT model π^{sft}. We then run alignment training using proximal policy optimization (PPO) with the length reward (10) and a KL regularization strength $\beta = 0.1$. Detailed experimental setup can be found in Appendix D.1.

Figure 2**(a)** demonstrates a consistent increase in the obtained length reward for both retrained (▲) and DeRa (■) models as we vary λ. Furthermore, the generated responses from both retrained and DeRa models exhibit a similar length distribution, as shown in Figure 2**(b)** and **(c)**. This suggests that DeRa can be used as a faithful approximation of the retrained model.

Although this simplified reward task serves as an illustrative example, it highlights the common reward-regularization tradeoff encountered in more realistic scenarios. Reward functions often target a specific subset of desired outcomes, making them susceptible to exploitation. In our toy example, the language model could potentially exploit the length reward function (10) by copying the query and truncating it anywhere within the length range $[L_{\text{min}}, L_{\text{max}}]$, thereby maximizing the reward. These responses, however, are not meaningful summaries. As corroborated by Figure 5 in the Appendix, the overall summarization quality deteriorates as the length reward increases. To mitigate reward hacking, one approach is to select an adequately large regularization strength using validation metrics such as automated evaluators or human evaluation. As we will demonstrate in subsequent experiments, DeRa facilitates the tuning of regularization strength without the need for retraining models.

5.3. Controlling alignment: a qualitative demonstration

We demonstrate DeRa's ability to control alignment during decoding with qualitative examples. We use Zephyr-7b models (Tunstall et al., 2023a), which are chat models fine-tuned based on the Mistral-7b model (Jiang et al., 2023). The checkpoints of SFT and aligned Zephyr-7b models are pub-

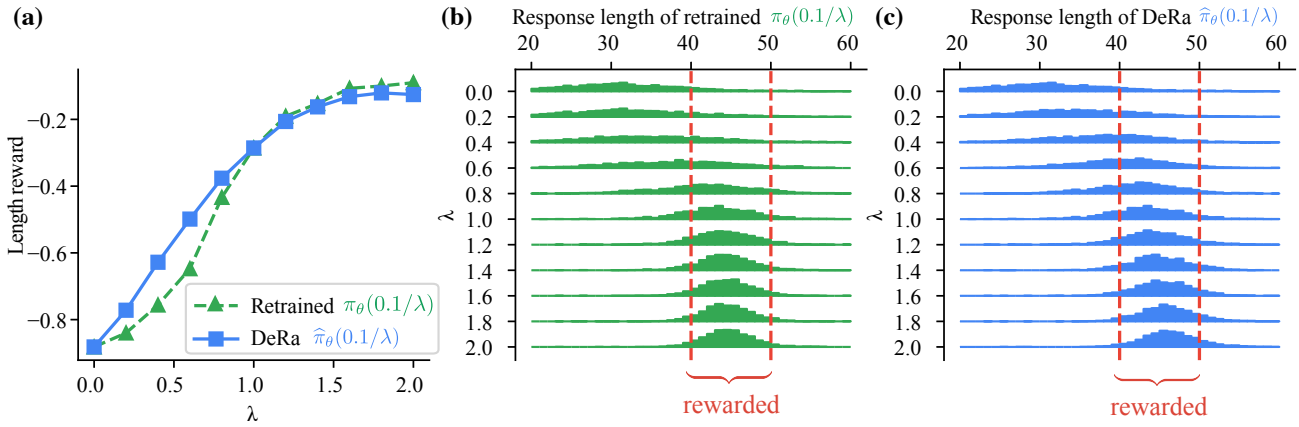

Figure 2. **Comparing DeRa and retrained models with different KL strengths in the length-reward task**. Panel (a): the length reward received by DeRa (■) and retrained models (▲) are comparable across different values of λ. Panel (b) and Panel (c): altering λ results in similar length distributions in both retrained models and DeRa models; the red dashed lines mark the rewarded range of $[40, 50]$.

licly available[1]. Specifically, as described in Tunstall et al. (2023a), the aligned Zephyr-7b model $\pi_\theta(\beta)$ with $\beta = 0.1$ was obtained based on the SFT model π^{sft} by training on binary preference samples from the UltraFeedback (Cui et al., 2023) dataset with DPO; see Tunstall et al. (2023a) for more details. With the SFT model π^{sft} and the aligned model π_θ, we use DeRa (Algorithm 1) to sample from different realigned models $\widehat{\pi}_\theta(\beta/\lambda)$. Figure 1 demonstrates the responses correspond to different realigned KL strength (β/λ), with $\lambda = 0, 1/6, 1, 5$, and 100. We demonstrate that adjusting the configurable λ in DeRa meaningfully controls the degree of alignment. While Figure 1 provides a qualitative example, further quantitative results are available in Appendix D.5. There, we apply DeRa to obtain various realigned Zephyr-7b models $\widehat{\pi}_\theta(\beta/\lambda)$ by linearly adjusting λ from 0 to 2.0 in increments of 0.2. We then evaluate these models using MT-Bench (Zheng et al., 2023), demonstrating that λ effectively controls the alignment level (Figure 8).

5.4. Learning to summarize from human feedback

We now tackle the more complex learning-to-summarize task using the Reddit TL;DR summarization dataset from Stiennon et al. (2020). Our goal is to empirically test whether DeRa can effectively guide the search for a suitable KL regularizing strength.

Obtaining SFT and aligned models. We first train a SFT model π^{sft} based on a pre-trained T5-Large model (Raffel et al., 2020), following the procedure in Stiennon et al. (2020) and Munos et al. (2023). Next, we train a separate T5-Large model to serve as a reward model using the preference dataset from Stiennon et al. (2020). Finally, we perform alignment training to optimize the expected reward, regularized with a KL strength of $\beta = 0.1$, using a policy gradient approach. This yields an aligned model $\pi_\theta(\beta)$ with $\beta = 0.1$. Experimental details are provided in Appendix D.3.

Apply DeRa. To investigate whether alternative KL strengths β/λ could outperform the base KL strength β, we apply DeRa by varying λ over a wide range of values $\{0.5, 2/3, 1.0, 2.0, 5.0, 10.0\}$, which corresponds to KL strengths of β/λ in the range $\{0.2, 0.15, 0.1, 0.05, 0.02, 0.01\}$. Note that the cases of $\lambda = 2.0, 5.0$, or 10.0 involve extrapolation, where the combined logits lie outside the convex hull of the reference and aligned logits. Our aim is to stress test these extrapolating λ values to assess whether DeRa can still provide a reasonable approximation when using these values.

Evaluating models. To evaluate the performance of DeRa models $\widehat{\pi}_\theta(\beta/\lambda)$ at different KL strengths β/λ, we use the highly capable PaLM 2 Large model (Anil et al., 2023) as a judge[2]. We extract 1000 queries from the summarization dataset's test fold and generate responses to these queries using each realigned model $\widehat{\pi}_\theta(\beta/\lambda)$. We then have the PaLM 2 model identify the better response from each pair of responses sampled from two different models. The win rate of each model against its counterpart is calculated by dividing the total number of samples preferred by the first model over the second model. These win rates are presented in Figure 3**(a)**, where the win rates of DeRa against the reference are indicated by ■. Additionally, the win rates of the retrained-from-scratch models $\pi_\theta(\beta/\lambda)$ against the SFT model π^{sft}, indicated by ▲, are evaluated as a benchmark. Similarly, Figure 3**(b)** compares the win rates of each model against the aligned model $\pi_\theta(\beta)$. Finally, Figure 3**(c)** show-

[1] https://huggingface.co/HuggingFaceH4/zephyr-7b-beta

[2] Specifically, we use the `text-unicorn-001` version.

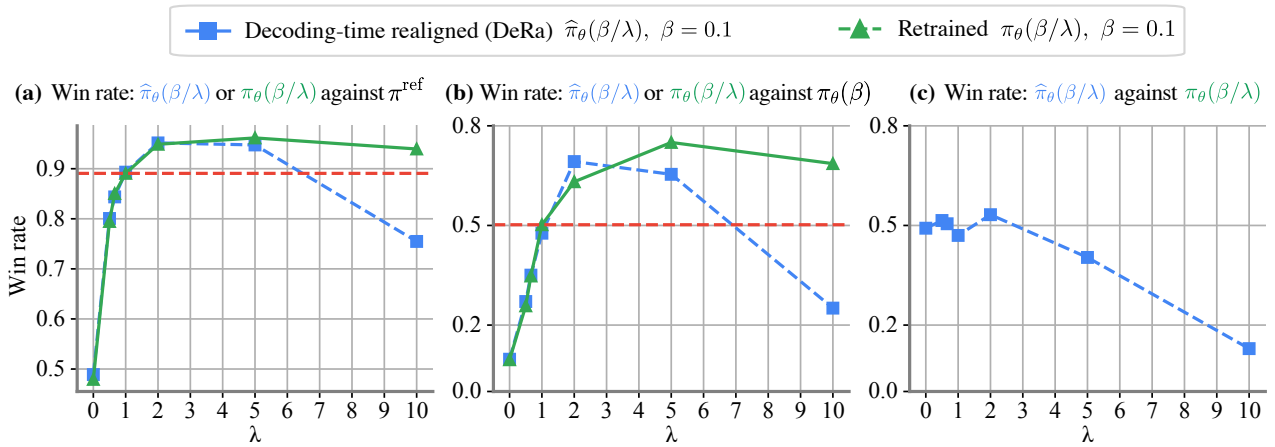

Figure 3. **Comparing DeRa and retrained models with different KL strengths in the summarization task**. Model π_θ are trained with the policy gradient method (see Appendix D.3). Panel (a): comparing DeRa models (■) or retrained model (▲) against the reference model. Panel (b): comparing DeRa (■) or retrained model (▲) against the base-aligned model. Panel (c): comparing DeRa against the retrained model. These results demonstrate that: (i) the performance of DeRa and retrained model is closely related, and (ii) DeRa enables the identification of KL strengths β/λ that outperform the original base KL strength β, whose performance is indicated by the red lines.

cases the win rates of sampling from DeRa $\widehat{\pi}_\theta(\beta/\lambda)$ against sampling from the retrained-from-scratch models $\pi_\theta(\beta/\lambda)$.

Findings. Figure 3 shows two key results. First, DeRa effectively identifies KL strengths β/λ that outperform the default KL strength β. These values are represented by the win rates above the red lines. Second, DeRa and retrained models generally agree well. Notably, this high agreement persists even with $\lambda > 1$, which are values in the extrapolation regime. This high agreement suggests that we can use DeRa as a cost-effective yet accurate method for determining effective KL strength hyperparameter.

Under- and over-regularization. In Figure 3, DeRa suggests that the aligned model $\pi_\theta(\beta)$ might be over-regularized. Reducing the KL regularization to β/λ for $\lambda > 1$ enhances the win rate, as confirmed by retrained models. In Figure 7 in the Appendix, we show that DeRa is also capable of identifying under-regularized models. In that case, DeRa recommends a greater KL strength (with a $\lambda < 1$) for improved performance.

5.5. Hallucination mitigation

To illustrate DeRa on another real-world problem, we provide a qualitative example of hallucination mitigation for Retrieval Augmented Generation (RAG; Lewis et al., 2020), popularized by the recent application of LMs to search engines such as Bing and Google. Specifically, our task is to rewrite a given list of pro and con arguments in natural prose. Importantly, the rewritten text must strictly adhere to the semantics of the original arguments, without introducing new content, that is, without hallucinating.

Experimental setup. We first train the PaLM 2 model (Anil et al., 2023) to act as a reward model using LoRA (Hu et al., 2022), then use a LoRA-based RLHF (Sun et al., 2023) regularized with a KL of strength $\beta = 0.1$ to align and mitigate hallucinations on a second PaLM 2[3]. We then apply DeRa by varying λ over $\{0.011, 0.1, 0.5, 0.67, 1, 2, 5, 10\}$, which corresponds to regularization strengths β/λ in the range $\{9, 1, 0.2, 0.15, 0.1, 0.05, 0.02, 0.01\}$. The details of the datasets used for the reward model and RLHF are presented in Appendix D.4.

Findings. The results are presented in Figure 4. Similarly to what was observed in Figure 1, we see that changes in λ lead to changes in the style of generation. More precisely, low values of λ lead to a behavior more similar to the reference model, and thus a higher tendency to hallucinate. As we increase λ, we improve the desired behavior of the model, where the arguments are correctly rewritten in natural prose without hallucinations. However, if when λ increases further, the naturalness of the language decreases, with the model resorting to copying and pasting the initial arguments verbatim, and losing coherence.

6. Related work

Multi-reward RLHF. Several studies have explored the issue of balancing multiple rewards (Rame et al., 2023; Jang et al., 2023; Mitchell et al., 2024). Unlike our approach, which combines a reference model and an aligned model, multi-reward methods aim to combine multiple models obtained from different rewards, each independently trained from the reference model. These methods are driven by

[3]Specifically, for this we use the `text-bison-001` version.

Arguments:
pro: Denying student loan debtors the benefits of bankruptcy--benefits that all other debtors have access to--is unfair.
con: Discharging student loan debt would only be a temporary bandage for the much larger problem of inflated college costs.

Response: λ=0.011
[...] it is unfair to deny student loan debtors the benefits of bankruptcy--benefits that all other debtors have access to. They also argue that student loan debt has a disproportionately negative impact on low-income borrowers [...]. However, [...] it would be a costly and ineffective solution to the problem of student debt. They also argue that it would only be a temporary bandage for the much larger problem of inflated college costs.

Response: λ=2
[...] One argument in favor of forgiveness is that it is unfair to deny student loan debtors the benefits of bankruptcy--benefits that all other debtors have access to. However, one argument against forgiveness is that it would only be a temporary bandage for the much larger problem of inflated college costs.

Response: λ=5
Denying student loan debtors the benefits of bankruptcy--benefits that all other debtors have access to--is unfair 301.\\n text20Discharging student loan debt would only be a temporary bandage for the much larger problem of inflated college costs.

Figure 4. **DeRa can control hallucinations in neutral response generation.** With a small λ (limited alignment), the sampled response includes hallucinations (highlighted in red), meaning semantic content not present in the argument provided. Increasing λ to 2 reduces hallucinations. However, at excessively high λ, the model begins to copy the argument verbatim (highlighted in gray), indicating reward hacking, and produces incoherent responses (highlighted in red and underlined).

the recognition that humans have diverse expectations when interacting with language models, each characterized by a reward function. For instance, in training chatbots to serve as human assistants, two pertinent reward functions are the chatbot's helpfulness and harmlessness. After training, these models are combined and weighted, either through parameter interpolation (Rame et al., 2023; Jang et al., 2023) or model ensemble (Mitchell et al., 2024), enabling to control the strength of each reward.

Proxy approaches for finetuning. Emulated fine-tuning (Mitchell et al. 2024; EFT) explores the capabilities of language models across two dimensions: model scale (large vs small models) and training stages (pretraining vs finetuning). EFT is a scale-decoupling approach that transfers the finetuning effects of a small LM to a large LM and vice versa. While Mitchell et al. (2024) used this approach as a tool for analyzing the capabilities of LMs, Liu et al. (2024a) demonstrated the empirical effectiveness of this proxy-tuning approach, showing it competes with standard finetuning across various benchmarks. Our approach shares similarities with EFT/proxy-tuning in that both merge trained models at the output level. While this approach decouples model scales, our objective is to strike a balance between reward and regularization. To that end, we show that our approach is a cost-effective proxy for full retraining with different degrees of alignment. Furthermore, Lu et al. (2023); Deng & Raffel (2023); Khanov et al. (2024); Huang et al. (2024) have explored using auxiliary models, such as separate reward models, to guide the text generation process of LMs.

Other decoding approaches that merge logits. Several sampling approaches merge logits from multiple language models like DeRa, but with different objectives. The fusion approaches (Gulcehre et al., 2015; Stahlberg et al., 2018) aim to use monolingual LMs to improve machine translation. Contrastive decoding approaches (Li et al., 2023) aim to enhance incoherence and lexical diversity in text generation. The Classifier-free guidance approach aims to improve prompt adherence in text generation. Speculative decoding approaches (Leviathan et al., 2023; Chen et al., 2023) speed up token generation by outputting multiple tokens at a time.

7. Conclusion

We introduced DeRa, a method for adjusting the KL regularization strength during decoding for realigning language models. Based on the variational perspective of the KL-regularized alignment objective in (1), we proved that aligned models with varying KL regularization strengths are all geometric mixtures; these mixtures combine the probability distributions of a reference model and an aligned model, varying only in their mixing weights. DeRa uses this knowledge to approximate these mixtures autoregressively during decoding. This approach gives DeRa a simple implementation and a clear interpretation.

One of DeRa's advantages is its ability to adjust regularization levels for individual users, prompts, or tasks. Many open-weights models, such as Llama-2 (Touvron et al., 2023) and Mistral 7B (Jiang et al., 2023), offer publicly available checkpoints for base and instruction-fine-tuned models. These models can act as references and aligned models alongside DeRa, allowing practitioners to tailor language model alignment to specific user preferences or downstream applications. Additionally, as we experimentally validated, DeRa can be used to efficiently identify promising regularization strengths to retrain a model. This streamlines hyperparameter tuning and reduces computational costs by avoiding unnecessary retraining across a wide range of regularization strengths.

Impact statement

We propose a method for exploring and adjusting regularization strengths in language model alignment. This work should be viewed within the broader context of language model alignment techniques, aiming to promote friendly, safe, and harmless responses in language models. It can also be viewed as a way to sweep over the regularization strength at decoding time, streamlining hyperparameter selection and reducing the number of retraining runs to get optimal regularization strengths.

Acknowledgment

We thank Johan Ferret for his helpful feedback on a draft of this paper. We thank Bilal Piot and Rémi Munos for their help with the auto-evaluator for the summarization task.

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. PaLM 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. Open LLM leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., Wang, T. T., Marks, S., Segerie, C.-R., Carroll, M., Peng, A., Christoffersen, P., Damani, M., Slocum, S., Anwar, U., Siththaranjan, A., Nadeau, M., Michaud, E. J., Pfau, J., Krasheninnikov, D., Chen, X., Langosco, L., Hase, P., Biyik, E., Dragan, A., Krueger, D., Sadigh, D., and Hadfield-Menell, D. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research (TMLR)*, 2023.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Proceedings of the Conference on Neural Information Processing Systems*, 2017.

Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., and Sun, M. UltraFeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Deng, H. and Raffel, C. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 11781–11791. Association for Computational Linguistics, 2023.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., and Bengio, Y. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Huang, J. Y., Sengupta, S., Bonadiman, D., Lai, Y.-a., Gupta, A., Pappas, N., Mansour, S., Kirchoff, K., and Roth, D. Deal: Decoding-time alignment for large language models. *arXiv preprint arXiv:2402.06147*, 2024.

Jang, J., Kim, S., Lin, B. Y., Wang, Y., Hessel, J., Zettlemoyer, L., Hajishirzi, H., Choi, Y., and Ammanabrolu, P. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Khanov, M., Burapacheep, J., and Li, Y. Alignment as reward-guided search. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Korbak, T., Perez, E., and Buckley, C. RL with KL penalties is better viewed as Bayesian inference. In *Proceedings*

of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2022.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *Proceedings of the International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2023.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Li, X. L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., and Lewis, M. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Liu, A., Han, X., Wang, Y., Tsvetkov, Y., Choi, Y., and Smith, N. A. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*, 2024a.

Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J., and Liu, J. Statistical rejection sampling improves preference optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024b.

Lu, X., Brahman, F., West, P., Jung, J., Chandu, K., Ravichander, A., Ammanabrolu, P., Jiang, L., Ramnath, S., Dziri, N., et al. Inference-time policy adapters (IPA): Tailoring extreme-scale lms without fine-tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6863–6883, 2023.

Mitchell, E., Rafailov, R., Sharma, A., Finn, C., and Manning, C. D. An emulator for fine-tuning large language models using small language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Munos, R., Valko, M., Calandriello, D., Azar, M. G., Rowland, M., Guo, D., Tang, Y., Geist, M., Mesnard, T., Michi, A., Selvi, M., Girgin, S., Momchev, N., Bachem, O., Mankowitz, D. J., Precup, D., and Piot, B. Nash learning from human feedback. *arXiv preprint*, 2023.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Pan, A., Bhatia, K., and Steinhardt, J. The effects of reward misspecification: Mapping and mitigating misaligned models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research (JMLR)*, 21(1):5485–5551, 2020.

Ramamurthy, R., Ammanabrolu, P., Brantley, K., Hessel, J., Sifa, R., Bauckhage, C., Hajishirzi, H., and Choi, Y. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Rame, A., Couairon, G., Shukor, M., Dancette, C., Gaya, J.-B., Soulier, L., and Cord, M. Rewarded soups: Towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Roberts, A., Chung, H. W., Levskaya, A., Mishra, G., Bradbury, J., Andor, D., Narang, S., Lester, B., Gaffney, C., Mohiuddin, A., Hawthorne, C., Lewkowycz, A., Salcianu, A., van Zee, M., Austin, J., Goodman, S., Soares, L. B., Hu, H., Tsvyashchenko, S., Chowdhery, A., Bastings, J., Bulian, J., Garcia, X., Ni, J., Chen, A., Kenealy, K., Clark, J. H., Lee, S., Garrette, D., Lee-Thorp, J., Raffel, C., Shazeer, N., Ritter, M., Bosma, M., Passos, A., Maitin-Shepard, J., Fiedel, N., Omernick, M., Saeta, B., Sepassi, R., Spiridonov, A., Newlan, J., and Gesmundo, A. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022. URL https://arxiv.org/abs/2203.17189.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized

advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Stahlberg, F., Cross, J., and Stoyanov, V. Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pp. 204–211. Association for Computational Linguistics, October 2018.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Sun, S., Gupta, D., and Iyyer, M. Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of RLHF. *arXiv preprint arXiv:2309.09055*, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., et al. Zephyr: Direct distillation of LM alignment. *arXiv preprint arXiv:2310.16944*, 2023a.

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rush, A. M., and Wolf, T. The alignment handbook. https://github.com/huggingface/alignment-handbook, 2023b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proceedings of the Conference on Neural Information Processing Systems*, volume 30, 2017.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. SLiC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Proof of Proposition 1

Proposition 1. The approximate realigned model $\widehat{\pi}_\theta(\beta/\lambda)$, defined in (6), can be equivalently written as

$$\widehat{\pi}_\theta(\beta/\lambda)(\cdot\,|x, y_{1:t-1}) = \texttt{softmax}\Big[\lambda \boldsymbol{h}_t^\theta(\beta) + (1-\lambda)\boldsymbol{h}_t^{\text{sft}}\Big]. \tag{9}$$

Proof of Proposition 1. We denote $\boldsymbol{p}_t^{\text{ref}} \coloneqq \pi^{\text{sft}}(\cdot|x, y_{1:t})$ and $\boldsymbol{p}_t^\theta \coloneqq \pi_\theta(\beta)(\cdot|x, y_{1:t})$. We note that

$$\boldsymbol{p}_t^{\text{ref}} \odot \left[\boldsymbol{p}_t^\theta \oslash \boldsymbol{p}_t^{\text{ref}}\right]^\lambda = \begin{bmatrix} \frac{\exp(\boldsymbol{h}_t^{\text{sft}}[1])}{\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\text{sft}}[i])} \\ \vdots \\ \frac{\exp(\boldsymbol{h}_t^{\text{sft}}[V])}{\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\text{sft}}[i])} \end{bmatrix} \odot \begin{bmatrix} \frac{\exp(\boldsymbol{h}_t^{\theta}[1])}{\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\theta}[i])} \\ \vdots \\ \frac{\exp(\boldsymbol{h}_t^{\theta}[V])}{\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\theta}[i])} \end{bmatrix}^\lambda \oslash \begin{bmatrix} \frac{\exp(\boldsymbol{h}_t^{\text{sft}}[1])}{\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\text{sft}}[i])} \\ \vdots \\ \frac{\exp(\boldsymbol{h}_t^{\text{sft}}[V])}{\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\text{sft}}[i])} \end{bmatrix}^\lambda \tag{11}$$

$$= \frac{1}{\Big(\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\text{sft}}[i])\Big)^{1-\lambda}\Big(\sum_{i=1}^V \exp(\boldsymbol{h}_t^{\theta}[i])\Big)^{\lambda}} \exp\Big(\lambda \boldsymbol{h}_t^\theta + (1-\lambda)\boldsymbol{h}_t^{\text{sft}}\Big), \tag{12}$$

where $\odot$ and $\oslash$ are entrywise product and division. It follows that

$$\widehat{\pi}_\theta(\cdot|x, y_{1:t}) = \frac{\boldsymbol{p}_t^{\text{ref}} \odot \left[\boldsymbol{p}_t^\theta \oslash \boldsymbol{p}_t^{\text{ref}}\right]^\lambda}{\left\|\boldsymbol{p}_t^{\text{ref}} \odot \left[\boldsymbol{p}_t^\theta \oslash \boldsymbol{p}_t^{\text{ref}}\right]^\lambda\right\|_1} = \frac{\exp\Big((1-\lambda)\boldsymbol{h}_t^{\text{sft}} + \lambda \boldsymbol{h}_t^\theta\Big)}{\left\|\exp\Big((1-\lambda)\boldsymbol{h}_t^{\text{sft}} + \lambda \boldsymbol{h}_t^\theta\Big)\right\|_1} = \texttt{softmax}\Big(\lambda \boldsymbol{h}_t^\theta + (1-\lambda)\boldsymbol{h}_t^{\text{sft}}\Big)$$

as claimed. □

B. Linear combination of multiple rewards

Following the notation of Section 2, we consider the case of a linear combination of rewards r_λ defined by

$$r_\lambda(x, y) = \sum_{i=1}^K \lambda_i r_i(x, y)\,,$$

for K reward functions r_i and $\lambda = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^K$.

Analogously writing $\pi^*(\beta, \lambda)$ for the realigned optimal distribution under the reward r_λ, we have that

$$\begin{aligned}\pi^*(\beta,\lambda)(y|x) &= \frac{\pi^{\text{sft}}(y|x)\exp[\frac{1}{\beta}r_\lambda(y|x)]}{\sum_{y'}\pi^{\text{sft}}(y'|x)\exp[\frac{1}{\beta}r_\lambda(y'|x)]} \\ &= \frac{\pi^{\text{sft}}(y|x)\exp[\frac{1}{\beta}\sum_{i=1}^K \lambda_i r_i(x,y)]}{\sum_{y'}\pi^{\text{sft}}(y'|x)\exp[\frac{1}{\beta}\sum_{i=1}^K \lambda_i r_i(x,y')]}\,.\end{aligned}$$

Denoting by $\pi_i^*(\beta)$ the optimal distribution realigned under the reward r_i, we have that

$$\frac{1}{\beta}r_i(x,y) = \log\frac{\pi_i^*(\beta)(y|x)}{\pi^{\text{sft}}(y,x)} + \log Z_i(x)\,.$$

Plugging this into the realigned distribution expression yields

$$\begin{aligned}\pi^*(\beta,\lambda)(y|x) &= \frac{\pi^{\text{sft}}(y|x)\prod_{i=1}^K\left(\frac{\pi_i^*(\beta)(y|x)}{\pi^{\text{sft}}(y,x)}Z_i(x)\right)^{\lambda_i}}{\sum_{y'}\pi^{\text{sft}}(y'|x)\prod_{i=1}^K\left(\frac{\pi_i^*(\beta)(y'|x)}{\pi^{\text{sft}}(y',x)}Z_i(x)\right)^{\lambda_i}} \\ &= \frac{\pi^{\text{sft}}(y|x)^{1-\bar\lambda}\prod_{i=1}^K \pi_i^*(\beta)(y|x)^{\lambda_i}}{\sum_{y'}\pi^{\text{sft}}(y'|x)^{1-\bar\lambda}\prod_{i=1}^K \pi_i^*(\beta)(y'|x)^{\lambda_i}}\,,\end{aligned}$$

where $\bar\lambda = \mathbf{1}^\top\lambda$ is the sum of the λ_i. Use of the logits $h_t^{\theta_i}(\beta)$ with multiple linear combinations (and autoregressive approximation) carries through in a similar fashion.

C. Alignment methods

Policy gradient methods The policy gradient method updates the LM with an estimate of the following gradient:

$$\mathbb{E}_{x\sim\rho, y\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta \log \pi_\theta(y|x)\Big(R(x,y) - \beta \mathrm{KL}\big(\pi_\theta(\cdot|x), \pi^{\mathrm{sft}}(\cdot|x)\big)\Big)\right]. \tag{13}$$

The proximal policy optimization (PPO; Schulman et al. 2017) is a variant of the vanilla policy gradient optimization. It replaces reward with general advantage estimation (Schulman et al., 2016) and introduces clipped probability ratios. For the length reward problem, we use PPO. For the summarization task, we used the vanilla policy gradient optimization (13), consistent with Munos et al. (2023).

Direct preference optimization Direct preference optimization (DPO; Rafailov et al., 2023) is an approach that directly optimizes the policy through a loss function defined via the Bradley-Terry reward model, without using a reward function. Given a dataset $\mathcal{D}$ that contains tuples of a query and two responses favored and unfavored by humans (x, y_w, y_l), the DPO loss is defined as

$$\mathcal{L}_{\mathrm{DPO}}\left(\pi_\theta; \pi^{\mathrm{sft}}\right) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta\left(y_w \mid x\right)}{\pi^{\mathrm{sft}}\left(y_w \mid x\right)} - \beta\log\frac{\pi_\theta\left(y_l \mid x\right)}{\pi^{\mathrm{sft}}\left(y_l \mid x\right)}\right)\right].$$

We use DPO in the summarization task (Appendix D.3) and in the chat model alignment task (Appendix D.5).

D. Details of experiment setup

D.1. Length-reward experiments

For this toy task of length reward, we use T5-small (Raffel et al., 2020) for policy and reward models. In supervised finetuning, we take a pretrained T5-small model (Roberts et al., 2022) and fine tune it on the Xsum dataset (Narayan et al., 2018) with 15k steps in a batch size of 32, yielding a model π^{sft}. With π^{sft} as an initialization, we train aligned policy models $\pi^*(\beta/\lambda)$ to maximize the length reward (10) using PPO. The policy learning rate is 5e-6, and the value function learning rate is 1e-5.

As shown in the main text, a lower KL regularization β/λ (i.e., with a greater λ), allows aligned models $\pi^*(\beta/\lambda))$ to gain more length reward (Figure 2). However, as shown in Figure 5, a greater KL strength (a smaller λ) retains a higher summarization quality. We measured the summarization quality in a way identical to our approach in Section 5.4: we prompt a highly capable Palm 2 model and ask it to compare the win rate of the aligned model against the SFT model; see Appendix D.3 for more details of the auto-evaluation setup.

D.2. Controlling alignment: a qualitative demonstration

For this experiment, we use the checkpoints of SFT and aligned Zephyr-7b models (Tunstall et al., 2023a). To save memory, we load both SFT and aligned checkpoints[4] with 4-bit quantization. We follow the Colab demo of Zephyr[5], and set `max_new_tokens=256`, `temperature=0.7`, `top_k=50`, and `top_p=0.95` for sampling.

D.3. Summarization experiments

Model training. For this experiment, we use a T5-Large model as the policy model, and a separate T5-Large model as the reward model, as in Munos et al. (2023). The supervised finetuning and policy gradient alignment setting mirrors the settings of Munos et al. (2023). For DPO alignment, we use $\beta = 0.1$, learning rate 3e-6, batch size 64, and 20k training steps.

Palm 2 evaluation. We use Palm 2 (the version called `text-unicorn-001`) to compare the quality of summarized responses. Given a to-be-summarized query ⟨`text`⟩ and two summaries ⟨`summary1`⟩ and ⟨`summary2`⟩ sampled from two models π_1 and π_2, we prompt the Palm 2 Large LM with:

```
'You are an expert summary rater.  Given a piece of text and two of its possible
```

[4] https://huggingface.co/HuggingFaceH4/zephyr-7b-beta

[5] https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha/blob/main/colab-demo.ipynb

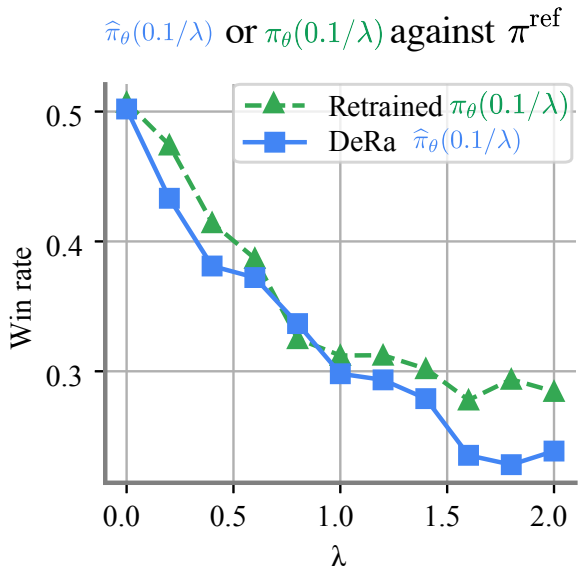

Figure 5. Comparing DeRa and retrained models against the SFT model using Palm 2 auto-evaluation; the win rate of both DeRa and retrained model decrease, since the length reward is an ineffective proxy of summarization quality.

```
summaries, output 1 or 2 to indicate which summary is better.  Text - ⟨text⟩,
Summary 1 - ⟨summary1⟩, Summary 2 - ⟨summary2⟩.  Preferred Summary -'
```

To avoid positional bias, we swap ⟨summary1⟩ and ⟨summary2⟩ with probability 0.5, and then consistently swap back the generated preferences of Palm 2. We then compute the ratio of the number of times that responses from the π_1 are preferred over than the second model π_2; we call this ratio the win rate of π_1 against π_2.

Pairwise acurracy evaluation. In addition to using the Palm 2 for evaluation, we also consider alternative evaluation metrics. For a given LM π, and a given pairwise dataset D be a dataset that contains tuples of a query and two responses favored and unfavored by humans ((x, y_w, y_l)), we let the pairwise accuracy be

$$\text{PairAcc}(\pi; D) = \mathbb{E}_{(x,y_w,y_l)\sim D}\, \mathbf{1}\Big(\frac{1}{|y_w|}\log \pi(y_w|x) > \frac{1}{|y_l|}\log \pi(y_l|x)\Big). \tag{14}$$

Intuitively, the pairwise accuracy (14) is high if the average log probability for the winning responses y_w is generally greater than that of the losing responses y_l. Figure 6 shows the pairwise accuracy of retrained models and DeRa models at various λ. The results of DeRa (■) and retrained models (▲) are overall close for λ being small; suggesting that DeRa is a sensible proxy for the retrained model. However λ increases, the gap between the performance of DeRa and the retrained model increases. This is expected, since $\lambda > 1$ is in the extrapolation region, where $\widehat{\pi}_\theta(\beta/\lambda)$ in (9) fails to be a good approximator of $\pi^*(\beta/\lambda)$.

DPO alignment. While we only reported policy gradient alignment result in the main text, here we report result from DPO alignment (Figure 7).

D.4. Hallucination mitigation

Training the reward model. The dataset used to train the reward model contains 1888 examples, which are split into training, validation, and evaluation datasets with 723, 242, and 223 examples respectively. Out of the examples in the training split, 388 contain hallucinations. Each example is a triple consisting of (i) a prompt with instructions to rewrite a given list of arguments in natural language (ii) the corresponding generation by the model (iii) a human annotated hallucination score (1 if the generation does not contain hallucinations and 0 if it does). The quality of the human annotations was checked

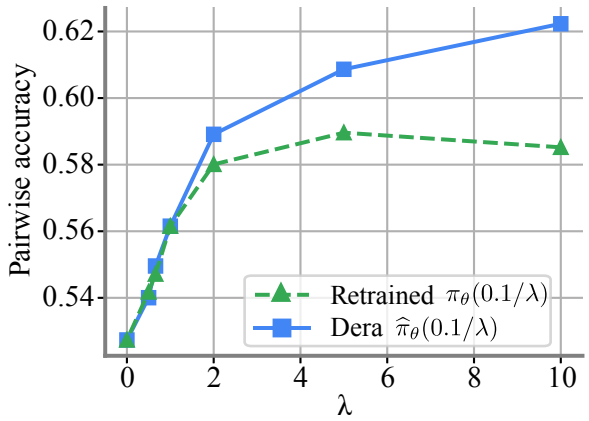

Figure 6. Comparing DeRa $\widehat{\pi}(0.1/\lambda)$ and retrained models $\pi_\theta(0.1/\lambda)$ with different KL strengths in the summarization task, using the pairwise accuracy metric.

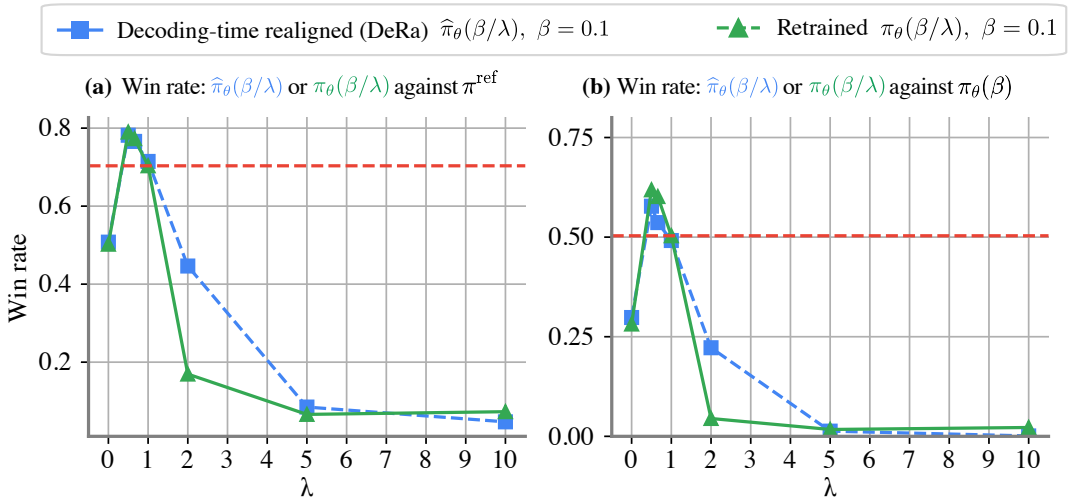

Figure 7. **Comparing DeRa $\widehat{\pi}(\beta/\lambda)$ and retrained models $\pi_\theta(\beta/\lambda)$ with different KL strengths in the summarization task**. All models π_θ are optimized with DPO. Panel (a): compare DeRa models $\widehat{\pi}_\theta(\beta/\lambda)$ (■) or retrained model $\pi_\theta(\beta/\lambda)$ (▲) against the reference model. Panel (b): compare DeRa $\widehat{\pi}_\theta(\beta/\lambda)$ (■) or retrained model $\pi_\theta(\beta/\lambda)$ (▲) against the base-aligned model $\pi_\theta(\beta)$. These comparisons show that performances of DeRa and the retrained model are correlated.

by a paid third-party pool of raters. Our final reward model acting as a classifier achieves an ROC-AUC of 0.986 on the evaluation set.

Details on RLHF. The dataset used to perform the RLHF contains 91 examples, which are split into training, validation, and evaluation datasets with 60, 20, and 11 examples respectively. This seemingly small dataset is compatible with the parameter-efficient tuning technique used (LoRA). Each example in these datasets consist of an user query on a sensitive question accompanied by a set of arguments in favor and against the topic in question, along with an expected "good" answer (without hallucinations). The temperature used for generations is set to $T = 1$ and we run RLHF for a thousand steps.

Prompt used for evaluation. To evaluate the resulting quality of the model after RLHF, we used a state of the art LLM as an auto-rater for our task. Our prompt uses two techniques, namely we provide six few-shot examples in it and we demand the model to execute the task as an expert. Each few-shot example has the following structure:

```
Reference arguments:  [ {arguments} ]
The natural language style paragraph version of these arguments:  [ {paragraph} ]
```

```
Expert review: the natural language paragraph contains additional points to the
reference arguments (yes/no):
```

The evaluation prompt is then constructed using the following template:

```
The following are examples of an expert noting when a natural language style
paragraph of text contains additional arguments to a given set of reference
arguments on a topic.

{fewshot example 1}

{fewshot example 2}

{fewshot example 3}

{fewshot example 4}

{fewshot example 5}

{fewshot example 6}

Expert review of an additional case where it was not initially known if it
contained extra arguments or not:

Reference arguments: [ {arguments} ]
The natural language style paragraph version of these arguments: [ {paragraph} ]
Expert review: the natural language paragraph contains additional points to the
reference arguments (yes/no):
```

D.5. Aligning general-purpose chat models

We apply DeRa to adjust the KL regularization strength for general-purpose chat models. We focus on Zephyr-7b (Tunstall et al., 2023a), a high-performing, open-weight chat model that is fine-tuned based on the Mistral 7b model (Jiang et al., 2023) with DPO. The open-weight checkpoints of the Zephyr-7b contain both the SFT model π^{sft}, and the aligned model $\pi_\theta(\beta)$ trained at the KL strength $\beta = 0.1$ (Tunstall et al., 2023a;b). With DeRa, we can flexibly explore the performance of Zephyr-7b models at different KL strengths.

We apply DeRa to obtain different realigned Zephyr-7b models $\widehat{\pi}_\theta(\beta/\lambda)$ by linearly sweeping λ from 0 to 2.0 with a stepsize 0.2. We then validate these models using MT-Bench (Zheng et al., 2023), a dataset with 160 questions across eight knowledge areas, formatted in a multi-turn style. The performance of different realigned models $\widehat{\pi}_\theta(\beta/\lambda)$, indicated by ■ in Figure 8, is evaluated based on the quality of its responses to these questions, with GPT-4 providing scores ranging from 1 to 10. Inspired by the rewarded soup (Rame et al., 2023) approach, we also evaluated a weight-combine variant of DeRa, whose performance is indicated by ● in Figure 8. While the standard DeRa (■) linearly combine logits (Algorithm 1), the weight-combining DeRa linearly combine parameters of the aligned model and the reference model, as in Rame et al. (2023).

Based on Figure 8, we anticipate a limited potential for improvement by adjusting KL regularization strength. This is because the base KL strength with $\lambda = 1$ pretty much attains the best MT-bench score in Figure 8. However, based on Figure 8, we hypothesize that another sensible choice of KL strength is around $\lambda = 0.5$. This is because the MT-bench score remains close to its peak at this value, and a lower λ leads to stronger KL regularization, which can be beneficial for tasks that rely on the SFT model's capabilities. One such task is mathematical reasoning, as alignment training generally diminishes models' performance on math reasoning problems like the GSM8k task (Beeching et al., 2023) of solving grade school math problems. To test this hypothesis, we re-train Zephyr-7b at $\lambda = 0.5$ or $\beta/\lambda = 0.2$. We confirm that its MT-bench score closely matches the original Zephyr model's score at with a KL strength 0.1. Additionally, we demonstrate that a stronger KL strength preserves the model's strong performance on math problems, leading to an overall higher score on the Open LLM leaderboard (Beeching et al., 2023), as shown in Table 1 in the Appendix.

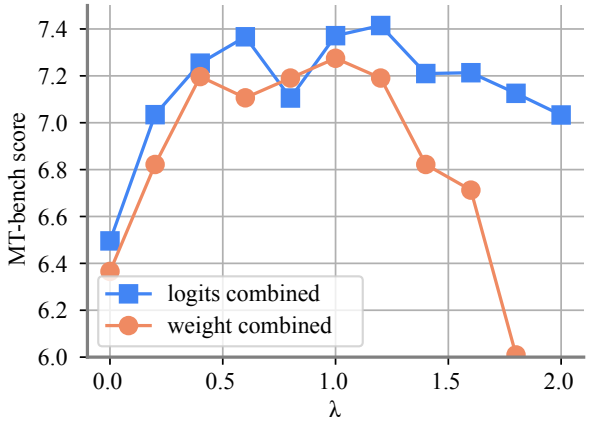

Figure 8. **Evaluating Zephyr-7b models with different DeRa-adjusted KL strengths on MT-bench**. These DeRa models $\widehat{\pi}_\theta(\beta/\lambda)$ use $\beta = 0.1$ and $\lambda \in \{0, 0.2, 0.4, \ldots, 2.0\}$ on MT-bench. The cases of $\lambda = 0$ and $\lambda = 1$ match the SFT model π^{sft} and aligned checkpoints $\pi_\theta(\beta)$ provided in the Zephyr-7b release (Tunstall et al., 2023a).

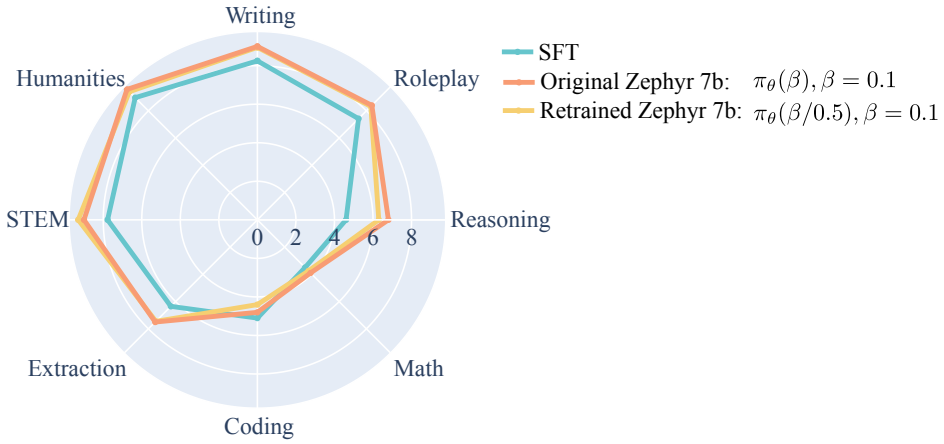

Figure 9. Comparing MT-bench scores of the original Zephyr 7b (Tunstall et al., 2023a) with KL strength 0.1 and the retrained Zephyr 7b at KL strength $0.1/0.5$. The results are very close. The average score of the original Zephyr 7b $\pi^*(\beta)$ with $\beta = 0.1$ is 7.37; the average score of the retrained model $\pi^*(\beta/\lambda)$ with $\beta = 0.1$ and $\lambda = 0.5$ is 7.23.

	Average	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	GSM8K
Zephyr-7b-beta-SFT	59.78	57.42	82.23	61.42	43.58	77.58	**36.47**
Zephyr-7b-beta [original; $\pi_\theta(0.1)$]	59.23	**62.03**	**84.36**	61.07	**57.45**	**77.74**	12.74
Zephyr-7b-beta [retrained; $\pi_\theta(0.1/0.5)$]	**61.55**	61.77	84.04	**61.79**	54.72	76.95	30.02

Table 1. Open LLM leaderboard.