KL-Regularised Q-Learning: A Token-level Action-Value perspective on Online RLHF

Jason R. Brown [*1] **Lennie Wells** [*2] **Edward James Young** [*3] **Sergio Bacallado** [2]

Abstract

Proximal Policy Optimisation (PPO) is an established and effective policy gradient algorithm used for Language Model Reinforcement Learning from Human Feedback (LM-RLHF). PPO performs well empirically but has a heuristic motivation and handles the KL-divergence constraint used in LM-RLHF in an ad-hoc manner. In this paper, we develop a a new action-value RL method for the LM-RLHF setting, KL-regularised Q-Learning (KLQ). We then show that our method is equivalent to a version of PPO in a certain specific sense, despite its very different motivation. Finally, we benchmark KLQ on two key language generation tasks—summarisation and single-turn dialogue. We demonstrate that KLQ performs on-par with PPO at optimising the LM-RLHF objective, and achieves a consistently higher win-rate against PPO on LLM-as-a-judge evaluations.

1. Introduction

LLMs are typically tuned through a three-stage process: generative pre-training, supervised fine tuning (SFT), and Language Model-Reinforcement Learning from Human Feedback (LM-RLHF) (Ouyang et al., 2022; Bai et al., 2022; Huang et al., 2024). These foundational works used the Proximal Policy Optimisation (PPO) algorithm for LM-RLHF, which is still widely used and viewed as the canonical algorithm for LM-RLHF (Xu et al., 2024).

A number of recent proposals have built upon PPO to develop alternative algorithms for the LM-RLHF setting, notably GRPO (Shao et al., 2024), which works with the completion-level rather than token-level MDP and uses group-rollouts to compute advantage estimates. There have also been a number of action-value methods proposed for LM-RLHF, but typically these work in the completion-level RL setting (Snell et al., 2023), or are offline RL methods. We give a more thorough literature review in Section 6.

By contrast, we sought to develop an *online action-value* method specifically tailored to the *token-level* LM-RLHF setting. We introduce our algorithm, KL-regularised Q-Learning (KLQ), in Section 3. KLQ exploits the discrete state-action space, and the presence of KL-regularisation against a base-policy. We use λ-returns to construct regressions targets and leverage a powerful analytic correspondence between optimal policies and action-values in the KL-regularised setting.

Following our presentation of KLQ, we go on to show analytically that the updates implemented by KLQ are equivalent to updates performed by (a modified version of) PPO, despite the loss function for KLQ being substantially simpler. This result was largely inspired by the seminal work of Schulman et al. (2018a).

Finally, we compare the performance of KLQ to PPO on two standard benchmarks: TL;DR for summarisation, and Anthropic-HH for single-turn dialogue. We demonstrate that KLQ optimises the LM-RLHF objective similarly well to PPO, with equal per-update compute cost. Moreover, we observe that the policy learned by KLQ outperforms the policy learned by PPO in direct LLM-as-a-judge evaluations.

We hope that these results will help the community improve its understanding of what is important for LM-RLHF, and open the door to future action-value approaches.

2. Background

We will suppose that the reader is familiar with standard Reinforcement Learning (RL) (Sutton & Barto, 2018) and with the three stage LLM pipeline of generative pre-training, supervised fine-tuning, and RLHF (Ouyang et al., 2022). Instead, we focus on the less common framework of KL-regularised RL, which is a key motivation for our method.

[*]Equal contribution [1]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK [2]Statistics Laboratory, Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, UK [3]Computational and Biological Learning Group, Department of Engineering, University of Cambridge, Cambridge, UK. Correspondence to: Edward James Young <ey245@cam.ac.uk>, Lennie Wells <ww347@cam.ac.uk>, Jason R. Brown <jrb239@cam.ac.uk>.

ICML 2025 Workshop on Models of Human Feedback for AI Alignment, Vancouver, Canada.

2.1. KL-regularised RL

KL-regularised RL involves sequential interactions between an agent and an environment. At time t, the agent finds itself in state s_t from a set $\mathcal{S}$ of possible states. The agent then samples an action a_t from some set $\mathcal{A}$ of possible actions according to a stochastic policy $\pi(\cdot|s)$. The environment then administers the agent some reward $r_{t+1} \in \mathbb{R}$ and transitions into a new state s_{t+1}. Let $\gamma \in (0, 1)$ be a discount factor and T be the (possibly infinite) time-step on which a terminal state is reached and the episode ends. Let π_b be a known (reliable) reference policy and $\tau > 0$ be a temperature parameter controlling the strength of the regularisation to this policy. Then the KL-regularised RL objective is to maximise the expectation of the discounted, KL-regularised return

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} \left(r_{k+1} - \tau\gamma D(\pi_\theta||\pi_b)\left(s_{k+1}\right)\right), \quad (1)$$

where $D(\pi_\theta||\pi_b)(s)$ denotes the KL divergence between the action distributions at state s:

$$D(\pi_\theta||\pi_b)(s) := \sum_a \pi_\theta(a|s) \log\left(\frac{\pi_\theta(a|s)}{\pi_b(a|s)}\right). \quad (2)$$

This KL-regularised RL setting can be viewed as generalizing that of entropy-regularised RL (Levine, 2018; Schulman et al., 2018a), which is more common in the online setting and for which a set of highly practical effective algorithms have been developed (Haarnoja et al., 2017; 2019). The theory of KL-regularised setting is a straightforward extension of the entropy-regularised theory. For completeness, we repeat core standard results in App. A.

Remark 2.1. In the case where $\gamma = 1$, the regularisation over states in Equation (1) is equivalent to a KL regularisation over full trajectories $(s_1, \ldots, s_T)$. In the LM-RLHF setting we will only ever take $\gamma = 1$, but we will present our algorithm for general γ to clarify how it fits in to this standard discounting framework.

2.2. LLMs and RLHF

We take the original RLHF pipeline for fine-tuning LLMs laid out in Ziegler et al. (2020) as our starting point. This trains LLMs for use in a *prompt-completion* setting. RLHF methods can be viewed either on the token-level or the completion level.

To fix notation, write $\mathcal{T}$ for space of all tokens and $\mathcal{T}^* = \cup_{n\in\mathbb{N}}\mathcal{T}^n$ for the space of finite sequences of tokens. We can then denote a language model as $\pi(a|s)$, where $a \in \mathcal{T}, s \in \mathcal{T}^*$. We represent a prompt by a sequence of tokens $x \in \mathcal{T}^*$. The model will need to return some completion $y \in \mathcal{T}^*$, by auto-regressive sampling up to some fixed length, or until a distinguished 'end-of-sequence' (EOS) token has been sampled. We will write $\pi(y|x)$ for the probability mass function corresponding to this auto-regressive generation.

Write $\pi_b(a|s)$ for the policy generated by the supervised fine-tuning stage. Write $R_\phi(x, y)$ for the *reward model* trained on human preferences to estimate the quality of a response y for a given query x. We will refer to the outputs $R_\phi(x, y)$ of the reward model as 'preference scores' from now on to avoid ambiguity with the notion of reward in the standard RL setting.

A reinforcement learning (RL) algorithm is then used to fine-tune the SFT model to generate completions that receive high preference scores. It is typical to regularise using the KL-divergence between the generation policy and a reference policy, typically obtained via SFT policy, and which we will denote by π_b for consistency with the KL-regularised RL setting. The modified objective is given by:

$$\bar{R}_\phi(x, y) = R_\phi(x, y) - \tau \log\left(\frac{\pi_\theta(y|x)}{\pi_b(y|x)}\right). \quad (3)$$

The KL-divergence penalty ensures that completions remain plausible under the SFT policy, avoiding mode-collapse and degenerate behaviour from reward model overoptimisation (Gao et al., 2022).

There are two distinct ways to view this learning problem from the state-action-reward framework of RL. Either:

1. **Completion-level view:** Treat queries x as states and completions y as actions, leading to a *contextual bandit* setting. In this case, the KL-divergence is over the entire completion y. This approach is taken by the RLOO algorithm (Ahmadian et al., 2024).

2. **Token-level view:** Treat partial completions as states, $s_t = (x, y_{1:t})$, and next tokens as actions, $a_t = y_{t+1}$. In this case, a reward equal to the weighted negative token-wise KL-divergence, $-\tau D(\pi_\theta||\pi_b)(s_{t+1})$, is administered at every time step. Since the reward model evaluates full completions, a final reward R_ϕ for the completion is given at the EOS token. This leads to a *Markov Decision Process (MDP)* viewpoint. We take this approach, in common with PPO.

For $\gamma = 1$, these views are equivalent, since

$$\begin{aligned}&\mathrm{KL}(\pi(y|x)||\pi_b(y|x)) \\ &\quad = \mathbb{E}\left[\sum_{t=0}^{T-1} \mathrm{KL}(\pi(a_t|s_t)||\pi_b(a_t|s_t))\right].\end{aligned}$$

With either view, the full completion y counts as an episode, and one can view the return of a trajectory y as resulting in a

total *return* $\bar{R}_\phi(x, y)$. Moreover, following Remark 2.1, this is equivalent to working in the KL-regularised framework with π_b as the reference policy. In the remainder of our exposition we will write π_b instead of π_b, since we are only interested in the LM-RLHF setting.

Proximal Policy Optimisation (PPO). The PPO algorithm trains a stochastic policy $\pi_\theta(a|s)$ and a value function $V_\theta(s)$, often implemented as two heads off a common neural network body. PPO is an on-policy algorithm which alternates between successive phases where experience is generated using the current policy $\pi_\theta(a|s)$ and then the policy and value function are updated. The loss for the value function is based on a squared error between the value function output and value estimates formed from rollout experience. In the typical version of PPO, PPO-clip, the policy is trained using a 'clipped' objective, which increases the probability of high advantage actions, but removes the incentive for it to deviate too much from the previous policy iterate, π_{old}. Letting

$$\rho_\theta(a, s) = \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)},$$

the objective takes the form

$$\min\left(\rho_\theta(a, s)\hat{A}(s, a), \text{clip}_{1-\epsilon}^{1+\epsilon}\left[\rho_\theta(a, s)\right]\hat{A}(s, a)\right), \quad (4)$$

where $\hat{A}(s, a)$ is the estimated advantage for taking action a in state s based on the current value function, and ϵ is the clipping ratio. For concreteness, we give complete pseudocode for PPO-clip in the LM-RLHF setting in Appendix B; we refer the reader to Huang et al. (2024) for a careful discussion of the algorithms implementation details. Another variant of PPO that will be of interest is PPO-penalty. This has a slightly different policy objective,

$$\mathbb{E}_{\pi_{\text{old}}}\left[\rho_\theta(a, s)\hat{A}(s, a) - \beta D(\pi_{\text{old}}||\pi_\theta)(s) - \tau D(\pi_\theta||\pi_b)(s)\right], \quad (5)$$

that incorporates a KL-penalty to the previous iterate instead of clipping.

3. The KLQ Algorithm

Our proposed algorithm, KLQ, is like PPO in that it is *on-policy* and consists of the following two phases:

1. Gather experiences from rollouts according to the current policy.

2. Updating the parameters of language model neural networks using minibatch gradients to optimise a certain objective.

However, whereas PPO uses a policy-gradient objective, KLQ uses the following ℓ^2 loss between the *action-value function* $Q_\theta(s, a)$ and value-estimates $\hat{G}$:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(Q_\theta(s, a) - \hat{G}\right)^2\right], \quad (6)$$

where the expectation is over state, action, value-estimate triples.

There are two core components of our algorithm to specify: the construction of $\hat{G}$, and the parametrisation of Q_θ. We construct the value estimates $\hat{G}$ using λ-returns, as explained in section Section 3.1. We construct Q_θ using a natural mapping between the policies, state-values and action-values in the KL-regularised setting, as explained in Section 3.2. The full pseudocode for the KLQ algorithm can be found in Appendix B.

3.1. λ-return value estimator

A major consideration for a value-based methodology in the language modeling setting is the choice of value estimator. Many prominent action-value methods (Haarnoja et al., 2017; Mnih et al., 2015) use one-step returns for their value estimate. However, one-step methods are inappropriate for the LM-RLHF setting, since the non-KL-regularisation portion of the reward signal (generated by the reward model, $R_\phi(x, y)$) is very sparse, being administered only at the final time-step of the trajectory. We therefore choose a value formulation based on the λ-return framework. This allows for the effective propagation of reward information over longer time horizons.

The λ-return (Sutton, 1988) is a weighted combination of n-step returns. The parameter λ acts as a *truncation rate*. When $\lambda = 1$, we recover the zero-bias, high-variance full return. When $\lambda = 0$ we recover the high-bias, low-variance one-step-return. Intermediate values of the truncation rate λ interpolate between these, allowing for an optimal bias-variance trade-off. PPO typically makes use of Generalised Advantage Estimation (GAE) (Schulman et al., 2018b) to form advantage estimates, which uses λ-returns formulated in terms of the state-value function in order to estimate action-values. By contrast, KLQ applies the λ-return framework to action-value functions, and to prepare for our theoretical results in Section 4, we present these in the *conservative* case. The conservative λ-returns $G_t^{\lambda,\alpha}$ corresponding to our action-value function Q_θ are defined via:

$$G_t^{\lambda,\alpha}[Q_\theta] = \alpha\sum_{k=t}^{T-1}(\lambda\gamma)^{k-t}\delta_k + Q_\theta(s_t, a_t), \quad (7)$$

where δ_t is the temporal-difference (TD) error for action-

value functions in the KL-regularised setting,

$$\delta_t = r_{t+1} + \gamma \left(\sum_a \pi_\theta(a|s_{t+1}) Q_\theta(s_{t+1}, a) - \tau D(s_{t+1}) \right) - Q_\theta(s_t, a_t), \quad (8)$$

which quantifies the difference between the predicted value of actions and a one-step bootstrapped approximation of value. For further background on TD errors and their theory, see Sutton (1988); Harutyunyan et al. (2016).

Remark 3.1. The parameter $\alpha \in (0, 1]$ scales the error term to bring the returns closer to the current action-value estimates. Thus, $\alpha \to 0$ is maximally conservative, while $\alpha \to 1$ recovers the standard λ-returns. We take $\alpha = 1$ in all our experiments, and only consider general α in our theoretical result.

Fitting the action-value function $Q_\theta(s, a)$ to the value-estimates $\hat{G}$ corresponds to a (partial) *policy improvement* step - see Theorem A.3 in Appendix A. In particular, it moves the action-value function $Q_\theta(s, a)$ towards the action-values of the policy π used to generate the rollouts.

3.2. Action-value decomposition

One key barrier for the use of action-value function methods in the language modeling setting is the initialisation of the Q-function. For LM-RLHF, policy gradient methods have an obvious advantage, in that the policy can be initialised as the output of a previous stage in a training pipeline (see Section 2.2). Without this initialisation, RL training for tasks in natural language would be completely infeasible. In order to use action-value methods for LM-RLHF, we develop a principled way to convert between a policy initialisation and an action-value initialisation.

General $(\pi, V) \leftrightarrow Q$ mapping In the KL-regularised setting, policy improvement can be performed by greedily maximising the one-step expected value minus KL-divergence penalty, *i.e.*, the new policy becomes the argmax of the following objective over π:

$$\mathbb{E}_\pi \left[Q(s, a) - \tau \log \left(\frac{\pi(a|s)}{\pi_b(a|s)} \right) \right]. \quad (9)$$

See Theorem A.5 and Corollary A.6 in Appendix A. Solving this optimisation problem yields the Boltzmann policy with respect to Q,

$$\pi[Q](a|s) = \pi_b(a|s) e^{(Q(s,a) - V[Q](s))/\tau}. \quad (10)$$

Here $V[Q](s)$ is the Boltzmann state-value function, which serves as a normalisation factor for the probability Boltzmann distribution,

$$V[Q](s) = \tau \log \left(\sum_a \pi_b(a|s) \exp(Q(s, a)/\tau) \right). \quad (11)$$

This relationship defines a clean mapping from action value functions, Q, to the corresponding (policy, state-value function) pairs *at optimality*. This mapping can be inverted to recover an action-value function Q from a (policy, state-value function) pair, (π, V) via

$$Q[\pi, V](s, a) = \tau \log \left(\frac{\pi(a|s)}{\pi_b(a|s)} \right) + V(s). \quad (12)$$

Applying the mapping to our method We leverage this mapping to define a convenient parametrisation of our action-value function, which enables us to move between Q-space and (π, V)-space seamlessly. In particular, we parametrise our action-value function using the policy, as:

$$Q_\theta(s, a) = \tau \log \left(\frac{\pi_\theta(a|s)}{\pi_b(a|s)} \right) + V_\theta(s). \quad (13)$$

The policy $\pi_\theta(a|s)$ is initialised as the SFT-policy, π_b, and the Boltzmann state-value function V is a randomly initialised value head added to the policy network.

Under this parametrization, computation of the TD-error Eq. (8) is particularly efficient. The TD-error now becomes:

$$\delta_t = r_{t+1} + \gamma V_\theta(s_{t+1}) - Q_\theta(s_t, a_t). \quad (14)$$

This follows from the fact that the log-probability term in the action-value function, Eq. (13), and the log-probabilities in the KL-divergence penalty, Eq. (2), exactly cancel.

Note that the policy component of the action-value function $\pi_\theta(a|s)$ is never trained by itself on any kind of policy objective. Instead, the action-value function defined by Eq. (13) is trained via the ℓ^2 loss, Eq. (6), with λ-return value estimates, Eq. (7). Because the action-value function is parametrised in terms of the policy, Eq. (13), training the action-value function automatically updates the policy. Furthermore, our parametrisation ensures that the policy is always Boltzmann with respect to the action-value function, Eq. (10). Accordingly, the policy improvement steps in KLQ happen *implicitly*, as a result of our parameterisation Eq. (13).

4. Equivalence between KLQ and PPO updates

In this section, we demonstrate that the updates performed by KLQ in Q-space are equivalent to the updates performed by a modified version of PPO in (π, V)-space.

We begin by considering the loss function used by the KLQ algorithm, 6. Define a sequence $(Q_k)_k$ by iteratively minimising this loss function, starting at some Q_0:

$$Q_{k+1} \leftarrow \arg\min_{Q} \mathbb{E}\left[\left(Q(s,a) - G^{\lambda,\alpha}[Q_k](s,a)\right)^2\right] \quad (15)$$

where we have made explicit the dependence of $G^{\lambda,\alpha}$ on the action-value function Q_k and the starting state-action pair.

We now construct update rules in (π, V)-space which lead to the equivalent sequence of iterates; we state this formally in Proposition 4.1, and give a proof in Appendix C.

We begin by defining advantage estimates via:

$$\hat{A}[\pi, V](s,a) := G^{\lambda}[\pi, V](s,a) - V(s) \quad (16)$$

We will take the sequence of policy iterates π_{k+1} to be the unique maximisers of the following objective (*cf.* Eq. (5)):

$$\mathbb{E}_{\pi_k}\left[\frac{\pi(a|s)}{\pi_k(a|s)}\hat{A}[\pi_k, V_k](s,a)\right] - \beta D(\pi||\pi_k) - \tau D(\pi||\pi_b) \quad (17)$$

where the KL-divergences are averaged over the state-visitation distribution of π_k. This policy objective can be seen as a modified version of PPO-penalty. Instead of the policy ratio clipping trick employed by PPO-Clip, this objective penalises the KL-divergence between the current policy and the previous iterate. However, it differs from standard PPO-penalty in that the KL-divergence is reverse. The previous iteration regularisation strength β which pre-multiplies this term is defined in terms of α and τ by:

$$\beta = \tau\left(\frac{1-\alpha}{\alpha}\right). \quad (18)$$

We take next iterate state-value function V_{k+1} to be the unique minimiser of the following loss:

$$\mathbb{E}\left[\left(V(s) - y(s,a)\right)^2\right], \quad (19)$$

where

$$y(s,a) = G^{\lambda,\alpha}[V_k, \pi_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right). \quad (20)$$

This differs from the standard state-value loss used for PPO in two ways. Firstly, it uses the α conservative λ-returns in place of the standard λ-return. Secondly, the KL-penalty term uses π_{k+1} rather than π_k.

Proposition 4.1. *The sequence $(Q_k)_k$ defined by Equation (15) corresponds via the mapping from Section 3.2 to the sequence $(\pi_k, V_k)_k$ defined above by optimising Equations (17) and (19).*

4.1. Discussion of theoretical result

To see the significance of these result, we note that if provided states are sampled on-policy, the objective in Equation (17) is identical to the PPO-penalty objective from Equation (5), except for the reversal of the KL-divergence between the new policy and the previous policy. Since soft action-value methods with conservative backup operators implicitly maximise this objective, we can interpret them as performing a variant of proximal policy optimisation, with Equation (18) providing a mapping between the degree of conservativeness α and the previous iterate regularisation strength β.

Additionally, this result allows us to reinterpret the roll of bootstrapping within proximal policy optimisation algorithms. While bootstrapping for action-value methods has its grounding in iterative policy evaluation procedures, bootstrapping for proximal policy optimisation algorithms serves solely as a variance reduction technique which introduces undesirable bias into the process. However, the equivalence relationship laid out above allows us to reinterpret bootstrapping to form advantage estimates as implicitly implementing iterative policy evaluation.

5. Experiments

We test KLQ against our baseline of PPO on two standard datasets used in existing works. The first is **TL;DR** (Syed et al., 2018), for *summarisation*, where the model must generate summaries of a Reddit posts. The second is Anthropic-**HH** (Bai et al., 2022), for *single-turn dialogue*, where the model must provide a final response to a conversation between a user and an assistant that is helpful and harmless.

We initialise our policy with supervised finetuned (SFT) versions of Pythia-1B (Biderman et al., 2023), specific to each dataset. A Pythia-1B based reward model is used for TL;DR, and a GPT2-large (Radford et al., 2019) based reward model is used for HH. We ran experiments using a customised fork of the TRL library (Werra et al., 2020), and inherited default hyperparameter values from the TRL implementation of PPO. Unless otherwise specified, each training run uses these default hyperparameters, and consists of 75,000 episodes, taking around 5 hours on 4 A100 GPUs. Appendix D contains further details of our training setup. We note that while our experiments are preliminary in scale, they serve as an important proof-of-concept for our theoretical insights.

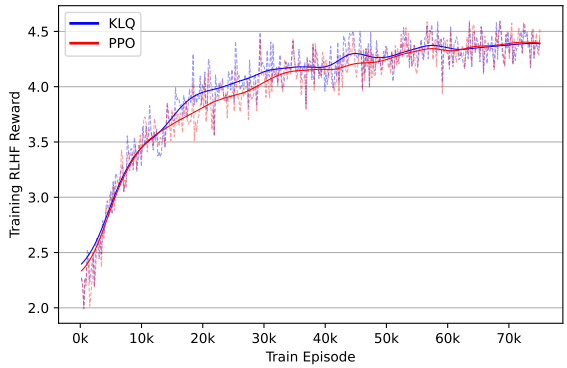
We first present representative training curves for KLQ and PPO in Section 5.1, which show that KLQ is competitive out-of-the-box.

We then present results from more extensive experiments to investigate the effect of varying the KL-penalty coefficient,

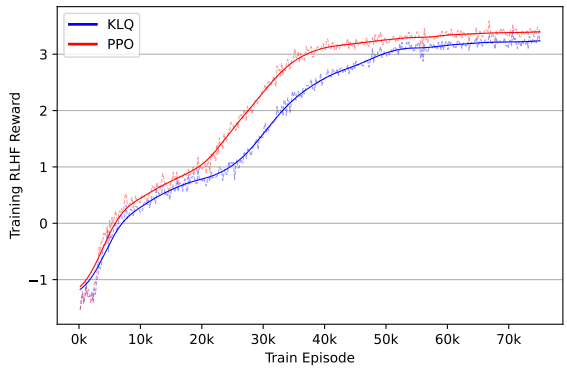
τ. Since KLQ incorporates KL-regularisation in a more sophisticated manner than PPO, we suspected there may be a different trade-off between reward and KL-divergence for different KL-penalty coefficients: we investigate the Pareto frontier of this trade-off in Section 5.2. We also suspected that there could be differences in qualitative aspects of our final completions: We explore this effect in Section 5.3 with LLM-as-a-Judge win-rates between the two models across the KL-penalty coefficients.

Additional ablation studies to understand the effects of varying learning rate and λ are presented in Appendices E.3 and E.4 respectively.

5.1. Training Curves



(a) TL;DR

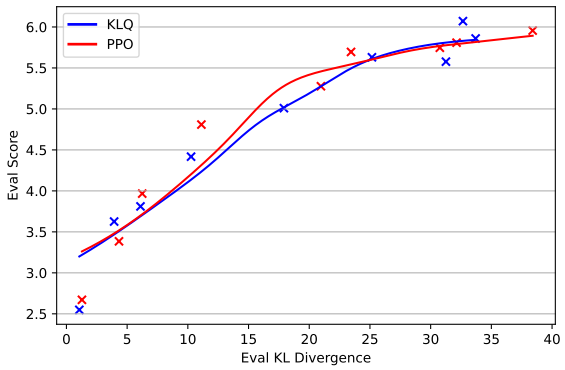


(b) HH

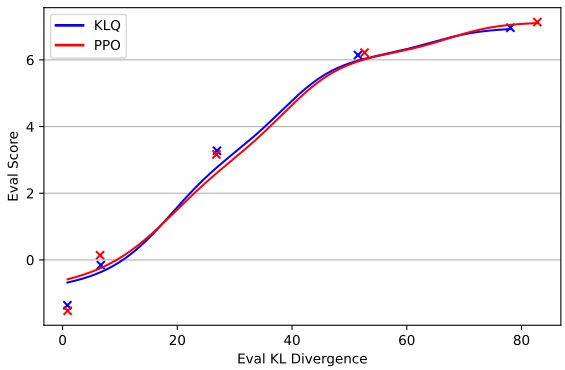
Figure 1. RLHF rewards over training for KLQ and PPO on (a) TL;DR and (b) HH. We observe similar performance between the two algorithms, especially with respect to the final reward achieved.

Figure 1 plots RLHF rewards over training for KLQ and PPO on TL;DR and HH. It shows KLQ and PPO achieving similar final results, with KLQ slightly lagging behind PPO at times on HH. The validation curves show similar behaviour and are given in Appendix E.1, along with training wall-clock times, which are consistent for both algorithms. This is particularly promising given that the default hyperparameters we used had previously been optimised for PPO, and we may expect significant improvements to KLQ from further hyperparameter tuning.

5.2. Pareto Frontier of Reward and KL-Divergence



(a) TL;DR



(b) HH

Figure 2. Relationship between evaluation reward model score and KL-divergence to the SFT policy at the end of training. Curves are plotted using Gaussian smoothing. Whilst there are some differences, they mostly appear to be due to noise.

Figure 2 illustrates the relationship between reward model score and KL-divergence to the SFT policy at the end of training, when evaluated on a held-out validation set. The KL-penalty coefficient, τ, was varied to generate a diverse range of samples. On both datasets we see very similar performance of the two algorithms. The lack of monotonicity in the plot also suggests that there is significant statistical noise in the training process. This is particularly clear from the corresponding training curves, which we plot in Appendix E.2.

5.3. LLM judges prefer KLQ

Figure 3 shows KLQ's win-rate against PPO on validation set prompt completions evaluated using GPT-4o mini (Ope-

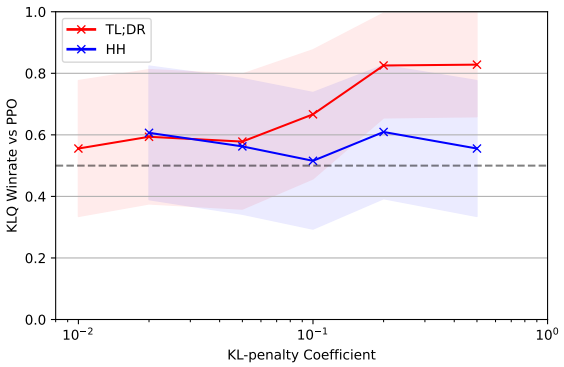


Figure 3. KLQ's win-rate vs PPO on validation set prompt completions evaluated using LLM-as-a-Judge, across various KL-penalty coefficients. Confidence interval shown is Jeffrey's Interval with $\alpha = 0.2$.

nAI, 2024) as a judge. We compare completions from the final models trained via each algorithm across a variety of different KL-penalty coefficients. For each dataset, we use a fixed set of 32 validation prompts. To enable Chain-of-Thought reasoning and aid interpretability, we prompt our judge to first compare the completions qualitatively before giving a preference; Appendix F displays the full prompts we used to instruct our judge. To control for ordering effects we present each prompt-completion pair to the judge in both orders (A vs B and B vs A); this gives a total of 64 queries per KL-penalty coefficient value per dataset.

Despite the similar performance of KLQ and PPO in Sections 5.1 and 5.2, here we see KLQ consistently outperforming PPO. While the confidence intervals are wide due to the limited number of test cases, KLQ maintains a higher win rate across all tested KL-penalty coefficients on both datasets, with particularly strong performance at higher coefficients on TL;DR. These results suggest that KLQ may achieve better generalization than PPO, producing higher-quality outputs while matching PPO on quantitative metrics. We hypothesise that this could be due to KLQ's more theoretically grounded handling of the KL-divergence constraint.

6. Related Work

6.1. Reinforcement learning (RL)

Entropy-regularised RL. The KL-regularised setting specialises to the more familiar entropy regularised setting by setting the reference policy to be an (unnormalised) constant policy. The entropy regularised setting has its roots in inverse RL (Ziebart et al., 2008), but has recently become very popular for continuous control with the SQL (Haarnoja et al., 2017) and SAC (Haarnoja et al., 2019) algorithms. Analogous versions of these algorithms have been developed for the discrete setting, namely Soft-DQN (Schulman et al., 2018a; Vieillard et al., 2020) and SAC-Discrete (Christodoulou, 2019) respectively. Of these, the closest to KLQ is Soft-DQN (Schulman et al., 2018a; Vieillard et al., 2020). KLQ differentiates itself from Soft-DQN in the following ways: KLQ is on-policy, while Soft-DQN is off-policy; KLQ utilises a special decomposition of the action-value function, Eq. (13); and KLQ uses λ-returns rather than simple one-step returns.

Action-value function decomposition The decomposition of the action-value function, Eq. (13), has appeared previously within the RL literature. Here we highlight two works which make use of it, and explain how we build on these.

Schulman et al. (2018a) demonstrates that, when the action-value function is decomposed in this way, gradient steps on the action-value loss are equivalent to a mixture of policy gradient steps (for the KL-regularised setting) and gradient steps on the state-value loss. This was a core inspiration for Section 4. However, in order for their equivalence to hold, new on-policy rollouts must be sampled after every gradient step on the action-value function. Accordingly, KLQ is not reducible to a simple policy gradient method, since we perform multiple epochs of minibatch gradient descent on the action-value loss, Eq. (6), for each new set of on-policy rollouts, as opposed to a single step. Additionally, Schulman et al. (2018a) is primarily a theoretical contribution, and does not propose a novel algorithm which relies upon this equivalence. Our work aims to fill this gap with KLQ.

Zhu et al. (2021) presents the VCWCV algorithm, which utilises a similar action-value decomposition to KLQ, with two key differences. First, VCWCV considers only the entropy regularised setting, taking the reference policy to be the (unnormalised) constant policy. Secondly, VCWCV is a continuous control algorithm, and parametrises the policy as a Beta distribution conditional on the state. This parametrisation places a significant restriction on the expressivity of the Q-function. By contrast, our decomposition in the discrete setting places no restrictions on the form of the Q-function. KLQ is further differentiated from VCWCV by its use of λ-returns and on-policy sampling.

λ-returns. The KLQ algorithm uses the λ-return action-value estimator, Eq. (7), with the TD-error given by Eq. (14). This estimator corresponds to the KL-regularised version of the on-policy $Q^\pi(\lambda)$ estimator (Harutyunyan et al., 2016) for policy evaluation. In fact, because our policy π is always Boltzmann with respect to the action-value function, our estimator also coincides with the KL-regularised version of the $Q^*(\lambda)$ estimator. Harutyunyan et al. (2016) demonstrated that, provided the sampling distribution μ is sufficiently close to the target policy π, the $Q^\pi(\lambda)$ estimator still de-

fines a valid contraction mapping with fixed point given by the target action-value function. This raises the possibility of an off-policy formulation of KLQ, which reuses previously sampled trajectories. However, since we are typically not limited by rollout generation in the language modeling setting, we decided not to make this design choice in KLQ.

6.2. RLHF

Group Rollouts. Various recent works have derived new algorithms for RLHF using grouped rollouts. By comparing preference scores given to different completions of the same prompt, it is possible to estimate advantages directly without the use of a value head. Some notable examples of this include: RLOO (Ahmadian et al., 2024), which apply the elegant REINFORCE Leave-One-Out procedure of Kool et al. (2019); GRPO (Shao et al., 2024), which was used for DeepSeekMath; and CoPG (Flet-Berliac et al., 2025), a very flexible recent proposal that uses a squared loss to align differences in policy log probabilities to differences in preference score. Note however that all these methods work in the completion-level setting and do not have an explicit method for token-level attribution.

DPO. KLQ leverages a conversion relationship between action-values and policies, Eq. (13), which allows us to train a policy using a value-based objective. Similarly, Direct Preference Optimisation (DPO) (Rafailov et al., 2024b) leverages a conversion relationship between reward models R_ϕ and policies in the contextual bandit setting to train a policy using a reward modelling objective. The key practical advantage of DPO (shared by subsequent preference optimisation techniques) is that it allows the policy π to be trained directly on the dataset of preferences, avoiding the need to use RL training. In interesting follow-up work, Rafailov et al. (2024a), show that DPO can in fact be interpreted as a form of Q-learning in the *token-level* MDP. Our work can also be interpreted as Q-learning in the token-level setting, and provides a further illustration that 'your language model is secretly an action-value function'.

7. Discussion

7.1. Off-policy flexibility

In common with PPO, KLQ is an *on-policy* algorithm, training only using rollouts generated from the current policy. Although policy-gradient methods can be made off-policy, doing so often entails significant complications (Wang et al., 2017; Espeholt et al., 2018) or a loss of theoretical justification (Lillicrap et al., 2019). In contrast, a core advantage of action-value methods is that they can retain their theoretical grounding regardless of whether data is sampled on- or off-policy (Haarnoja et al., 2017; Mnih et al., 2015). In future work we hope to explore the use of off-policy sampling as an extension to KLQ. In particular, off-policy data can include expert demonstrations—obviating the need for an SFT stage altogether—or prioritised experience replay (Schaul et al., 2016), allowing the reuse of trajectories that were particularly insightful.

7.2. State-value head learning

Recent algorithms that use group rollouts do not need to add a state-value head to their language model for LM-RLHF. Ahmadian et al. (2024) argue that the key benefit of this technique is to eliminate the bias introduced from bootstrapping using value estimates. Our initial λ-ablation experiments (see App. E.4) suggest a benefit to bootstrapping with a value head, at least for the action-value learning case with a single completion per prompt.

The value head is an essential component of our KLQ method, and we suspect there may be benefits of explicit token-level attribution. In future work we hope to investigate ways to combine this with the benefits of group-rollouts.

7.3. Advantage normalisation

The TRL implementation of PPO uses advantage whitening, and claims that this is important. It is not clear how to establish an analogue of this procedure for KLQ due to the combined π, V objective. There may be significant empirical improvements from establishing an appropriate whitening procedure for KLQ.

7.4. Limitations

The main limitation with this work at present is that our experiments are relatively preliminary compared to the most recent works in the LM-RLHF space. Firstly, each training run took around 5 hours on 4 A100 GPUs; we did not have the computational resources to perform longer experiments, large hyper-parameter sweeps, or many repeats to reduce the effects of noise. Secondly, we used existing SFT and reward models from the Huggingface Hub. Whilst we are thankful of the community and the open-access models, it was hard to evaluate or guarantee the quality and properties of these models. A more robust evaluation pipeline would involve training our own SFT and reward models tailored to our specific needs. We would also be excited to evaluate our methodology on multi-step reasoning tasks, which are currently an exciting application area for language model RL. Finally, there are certain aspects of the equivalence in

8. Conclusion

We propose KLQ, a novel algorithm for Language Model-Reinforcement Learning from Human Feedback (LM-RLHF). We benchmark our method against Proximal Policy

Optimisation (PPO), a canonical algorithm from previous works. KLQ has a cleaner theoretical motivation than PPO, performs similarly at optimising the LM-RLHF objective, and generates completions that are often preferred in LLM-as-a-judge evaluations.

We provide an analytic argument to show that optimising the KLQ objective is equivalent to optimising a modified version of the PPO objective, in a certain specific sense. This demonstrates a heretofore unappreciated connection between proximal policy optimisation algorithms and action-value methods. KLQ's clean motivation and close theoretical connection to PPO helps give an insight into the learning process of LM-RLHF, and points towards deeper reasons as to why PPO is effective.

Author Contributions

JB, LW, and EY spent an equal number of hours on the project, and contributed to writing code, running empirics and writing the manuscript. EY initially conceived the project, the KLQ algorithm and developed all the theoretical results. SB provided advice and guidance throughout the project.

Acknowledgements

EY and JB would like to thank their PhD supervisors Yashar Ahmadian and Robert Mullins for their guidance over the project, and providing access to compute resources. All authors would like to thank Usman Anwar for feedback on early iterations of the work, and to the Meridian office for providing a venue for in-person collaboration.

Impact Statement

This goal of this paper is to develop more effective fine-tuning procedures for large language models that produce more preferable completions. This goal should lead to more aligned models, which is clearly in the public interest; however, this could in turn lead to more powerful models being deployed, increasing the corresponding societal impacts and risks.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs, February 2024. URL http://arxiv.org/abs/2402.14740. arXiv:2402.14740 [cs].

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, April 2022. URL http://arxiv.org/abs/2204.05862. arXiv:2204.05862 [cs].

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., and others. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Christodoulou, P. Soft Actor-Critic for Discrete Action Settings, October 2019. URL http://arxiv.org/abs/1910.07207. arXiv:1910.07207 [cs].

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, June 2018. URL http://arxiv.org/abs/1802.01561. arXiv:1802.01561 [cs].

Flet-Berliac, Y., Grinsztajn, N., Strub, F., Wu, B., Choi, E., Cremer, C., Ahmadian, A., Chandak, Y., Azar, M. G., Pietquin, O., and Geist, M. Contrastive Policy Gradient: Aligning LLMs on sequence-level scores in a supervised-friendly fashion, January 2025. URL http://arxiv.org/abs/2406.19185. arXiv:2406.19185 [cs].

Gao, L., Schulman, J., and Hilton, J. Scaling Laws for Reward Model Overoptimization, October 2022. URL http://arxiv.org/abs/2210.10760. arXiv:2210.10760 [cs].

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement Learning with Deep Energy-Based Policies, July 2017. URL http://arxiv.org/abs/1702.08165. arXiv:1702.08165 [cs].

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft Actor-Critic Algorithms and Applications, January 2019. URL http://arxiv.org/abs/1812.05905. arXiv:1812.05905 [cs].

Harutyunyan, A., Bellemare, M. G., Stepleton, T., and Munos, R. Q($$) with Off-Policy Corrections, August 2016. URL http://arxiv.org/abs/1602.04951. arXiv:1602.04951 [cs].

Huang, S., Noukhovitch, M., Hosseini, A., Rasul, K., Wang, W., and Tunstall, L. The N+ Implementation Details of RLHF with PPO: A Case Study on TL;DR Summarization, March 2024. URL http://arxiv.org/abs/2403.17031. arXiv:2403.17031.

Kool, W., Hoof, H. v., and Welling, M. Buy 4 REINFORCE Samples, Get a Baseline for Free! April 2019. URL https://openreview.net/forum?id=r1lgTGL5DE.

Levine, S. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review, May 2018. URL http://arxiv.org/abs/1805.00909. arXiv:1805.00909 [cs].

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, July 2019. URL http://arxiv.org/abs/1509.02971. arXiv:1509.02971 [cs].

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236. Publisher: Nature Publishing Group.

OpenAI. GPT-4o System Card, October 2024. URL http://arxiv.org/abs/2410.21276. arXiv:2410.21276 [cs].

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, 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, March 2022. URL http://arxiv.org/abs/2203.02155. arXiv:2203.02155 [cs].

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. 2019.

Rafailov, R., Hejna, J., Park, R., and Finn, C. From r to Q^*: Your Language Model is Secretly a Q-Function, August 2024a. URL http://arxiv.org/abs/2404.12358. arXiv:2404.12358 [cs].

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, July 2024b. URL http://arxiv.org/abs/2305.18290. arXiv:2305.18290 [cs].

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized Experience Replay, February 2016. URL http://arxiv.org/abs/1511.05952. arXiv:1511.05952 [cs].

Schulman, J., Chen, X., and Abbeel, P. Equivalence Between Policy Gradients and Soft Q-Learning, October 2018a. URL http://arxiv.org/abs/1704.06440. arXiv:1704.06440 [cs].

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation, October 2018b. URL http://arxiv.org/abs/1506.02438. arXiv:1506.02438 [cs].

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, April 2024. URL http://arxiv.org/abs/2402.03300. arXiv:2402.03300 [cs].

Snell, C., Kostrikov, I., Su, Y., Yang, M., and Levine, S. Offline RL for Natural Language Generation with Implicit Language Q Learning, May 2023. URL http://arxiv.org/abs/2206.11871. arXiv:2206.11871 [cs].

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988. ISSN 1573-0565. doi: 10.1007/BF00115009. URL https://doi.org/10.1007/BF00115009.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction, 2nd ed*. Reinforcement learning: An introduction, 2nd ed. The MIT Press, Cambridge, MA, US, 2018. ISBN 978-0-262-03924-6. Pages: xxii, 526.

Syed, S., Voelske, M., Potthast, M., and Stein, B. Dataset for generating TL;DR, February 2018. URL https://zenodo.org/records/1168855.

Vieillard, N., Pietquin, O., and Geist, M. Munchausen Reinforcement Learning, November 2020. URL http://arxiv.org/abs/2007.14430. arXiv:2007.14430 [cs].

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and Freitas, N. d. Sample Efficient Actor-Critic with Experience Replay, July 2017. URL http://arxiv.org/abs/1611.01224. arXiv:1611.01224 [cs].

Werra, L. v., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. TRL: Transformer Reinforcement Learning, 2020. URL https://github.com/huggingface/trl. Publication Title: GitHub repository.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. v., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. HuggingFace's Transformers: State-of-the-art Natural Language Processing, July 2020. URL http://arxiv.org/abs/1910.03771. arXiv:1910.03771 [cs].

Xu, S., Fu, W., Gao, J., Ye, W., Liu, W., Mei, Z., Wang, G., Yu, C., and Wu, Y. Is DPO Superior to PPO for LLM Alignment? A Comprehensive Study, April 2024. URL https://arxiv.org/abs/2404.10719v2.

Zhu, J., Zhang, H., and Pan, Z. Value-Based Continuous Control Without Concrete State-Action Value Function. In Tan, Y. and Shi, Y. (eds.), *Advances in Swarm Intelligence*, pp. 352–364, Cham, 2021. Springer International Publishing. ISBN 978-3-030-78811-7. doi: 10.1007/978-3-030-78811-7_34.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, pp. 1433–1438, Chicago, Illinois, July 2008. AAAI Press. ISBN 978-1-57735-368-3.

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, January 2020. URL http://arxiv.org/abs/1909.08593. arXiv:1909.08593 [cs].

A. Theoretical Background

In this appendix we recount core theoretical results for the KL-regularised setting and λ-returns. These results follow straightforwardly from the analogous results in the entropy-regularised setting, so do not constitute a major contribution of our work. We include them here only for clarity of presentation and completeness of exposition. All results in this section apply to a finite MDP, *i.e.*, $|\mathcal{S} \times \mathcal{A}| < \infty$ with a discount factor $\gamma \in [0, 1)$ and rewards bounded above by $R_{\max}$. We always assume that $\lambda \in [0, 1]$.

The *soft action-value function* for a policy π, $Q^\pi(s, a)$ is defined by:

$$Q^\pi(s,a) := \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=t}^{T-1} \gamma^{k-t} \left(r_{k+1} - \tau\gamma D_\pi \left(s_{k+1} \right) \right) \middle| s_t = s, a_t = a \right] \tag{21}$$

where G_t is the KL-augmented return, Eq. (1), and $D_\pi(s)$ is the KL-divergence between π and π_{SFT} at s. We next present policy evaluation and improvement theorems for this setting.

Lemma A.1 (Bellman recursion for the soft action-value function)**.** *Let B^π be the soft Bellman operator for π, which acts on action-value functions via:*

$$\left[B^\pi Q \right](s,a) = \mathbb{E}_\pi \left[r_{t+1} + \gamma \left(Q(s_{t+1}, a_{t+1}) - \tau D_\pi \left(s_{t+1} \right) \right) | s = s_t, a = a_t \right]. \tag{22}$$

Then the soft action-value function, $Q^\pi(s, a)$, satisfies the Bellman recursion relationship, $B^\pi Q^\pi = Q^\pi$.

Proof of Lemma A.1. We begin with the definition of the soft action-value function:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a \right] && (23) \\ &= \mathbb{E}_\pi \left[\sum_{k=t}^{T-1} \gamma^{k-t} \left(r_{k+1} - \tau\gamma D_\pi \left(s_{k+1} \right) \right) \middle| s_t = s, a_t = a \right] && (24) \end{aligned}$$

Splitting the sum to separate the immediate reward and future returns:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[r_{t+1} - \tau\gamma D_\pi(s_{t+1}) + \gamma \sum_{k=t+1}^{T-1} \gamma^{k-(t+1)} \left(r_{k+1} - \tau\gamma D_\pi \left(s_{k+1} \right) \right) \middle| s_t = s, a_t = a \right] \tag{25}$$

The second term inside the expectation is precisely the KL-augmented return from time $t + 1$, denoted as G_{t+1}. By the definition of the soft action-value function, $\mathbb{E}_\pi \left[G_{t+1} | s_{t+1}, a_{t+1} \right] = Q^\pi(s_{t+1}, a_{t+1})$. Therefore:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi \left[r_{t+1} - \tau\gamma D_\pi \left(s_{t+1} \right) + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a \right] && (26) \\ &= \mathbb{E}_\pi \left[r_{t+1} + \gamma \left(Q^\pi(s_{t+1}, a_{t+1}) - \tau D_\pi \left(s_{t+1} \right) \right) | s_t = s, a_t = a \right] && (27) \\ &= \left[B^\pi Q^\pi \right](s,a) && (28) \end{aligned}$$

This confirms that Q^π satisfies the Bellman recursion relationship, $B^\pi Q^\pi = Q^\pi$. □

Theorem A.2 (KL-regularised policy evaluation)**.** *The soft Bellman operator is a contraction mapping in the ℓ^∞ norm with contraction modulus γ. Accordingly, the soft action-value function $Q^\pi(s, a)$ is the unique fixed point of B^π, and any sequence of iterates converges to $Q^\pi(s, a)$ in the ℓ^∞ norm.*

Proof of Theorem A.2. To prove that B^π is a contraction mapping in the ℓ^∞ norm with contraction modulus γ, we need to show that for any two action-value functions Q_1 and Q_2:

$$\| B^\pi Q_1 - B^\pi Q_2 \|_\infty \leq \gamma \| Q_1 - Q_2 \|_\infty \tag{29}$$

Let us expand the difference for any state-action pair (s, a):

$$\begin{aligned} &[B^\pi Q_1](s,a) - [B^\pi Q_2](s,a) \\ &= \mathbb{E}_\pi \left[r_{t+1} + \gamma \left(Q_1(s_{t+1}, a_{t+1}) - \tau D_\pi \left(s_{t+1} \right) \right) | s = s_t, a = a_t \right] \\ &- \mathbb{E}_\pi \left[r_{t+1} + \gamma \left(Q_2(s_{t+1}, a_{t+1}) - \tau D_\pi \left(s_{t+1} \right) \right) | s = s_t, a = a_t \right] \end{aligned} \tag{30}$$

The reward terms r_{t+1} and KL divergence terms $D_\pi(s_{t+1})$ cancel out, leaving:

$$|[B^\pi Q_1](s,a) - [B^\pi Q_2](s,a)| = \gamma \left|\mathbb{E}_\pi \left[Q_1(s_{t+1},a_{t+1}) - Q_2(s_{t+1},a_{t+1})|s=s_t, a=a_t\right]\right| \tag{31}$$

By Jensen's inequality, $|\mathbb{E}[X]| \leq \mathbb{E}[|X|]$, we have:

$$|[B^\pi Q_1](s,a) - [B^\pi Q_2](s,a)| \leq \gamma \mathbb{E}_\pi \left[|Q_1(s_{t+1},a_{t+1}) - Q_2(s_{t+1},a_{t+1})||s=s_t, a=a_t\right] \tag{32}$$

Since $\|Q_1 - Q_2\|_\infty = \max_{s,a} |Q_1(s,a) - Q_2(s,a)|$, we know that for any state-action pair (s_{t+1}, a_{t+1}):

$$|Q_1(s_{t+1},a_{t+1}) - Q_2(s_{t+1},a_{t+1})| \leq \|Q_1 - Q_2\|_\infty \tag{33}$$

This gives us:

$$\begin{aligned} |[B^\pi Q_1](s,a) - [B^\pi Q_2](s,a)| &\leq \gamma \mathbb{E}_\pi \left[\|Q_1 - Q_2\|_\infty | s=s_t, a=a_t\right] &(34)\\ &= \gamma \|Q_1 - Q_2\|_\infty &(35)\end{aligned}$$

Since this inequality holds for all state-action pairs (s,a), we have:

$$\|B^\pi Q_1 - B^\pi Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty \tag{36}$$

This proves that B^π is a contraction mapping with contraction modulus $\gamma < 1$. By the Banach fixed-point theorem, B^π has a unique fixed point, and any sequence of iterates $Q_{n+1} = B^\pi Q_n$ converges to this fixed point in the ℓ^∞ norm.

We've already shown in Lemma A.1 that Q^π is a fixed point of B^π. Therefore, Q^π is the unique fixed point of B^π, and any sequence of iterates converges to Q^π. □

The λ-returns used to formulate the value estimates for our action-value function loss $\mathcal{L}(\theta)$ are related to the λ soft Bellman operator, defined by:

$$B^\pi_\lambda = (1-\lambda) \sum_{n \geq 1} \lambda^{n-1} \left[B^\pi\right]^n. \tag{37}$$

In particular,

$$\left[B^\pi_\lambda Q\right](s,a) = \mathbb{E}_\pi \left[G^\lambda_t | s_t = s, a_t = a\right] \tag{38}$$

Note that the λ soft Bellman operator is a weighted average of n-step Bellman operators, $[B^\pi]^n$, with weights $(1-\lambda)\lambda^{n-1}$.

Theorem A.3 (λ policy evaluation)**.** *The λ soft Bellman operator, B^π_λ, is a contraction mapping in the ℓ^∞ norm with contraction modulus*

$$\gamma \left(\frac{1-\lambda}{1-\lambda\gamma}\right).$$

Accordingly, the soft action-value function $Q^\pi(s,a)$ is the unique fixed point of B^π_λ, and any sequence of iterates converges to $Q^\pi(s,a)$ in the ℓ^∞ norm.

Proof of Theorem A.3. For any two action-value functions Q_1 and Q_2, we have:

$$\|B^\pi_\lambda Q_1 - B^\pi_\lambda Q_2\|_\infty = \left\| (1-\lambda) \sum_{n \geq 1} \lambda^{n-1} \left(\left[B^\pi\right]^n Q_1 - \left[B^\pi\right]^n Q_2\right) \right\|_\infty \tag{39}$$

Using the triangle inequality:

$$\|B^\pi_\lambda Q_1 - B^\pi_\lambda Q_2\|_\infty \leq (1-\lambda) \sum_{n \geq 1} \lambda^{n-1} \left\|\left[B^\pi\right]^n Q_1 - \left[B^\pi\right]^n Q_2\right\|_\infty \tag{40}$$

From Theorem A.2, we know that B^π is a contraction mapping with modulus γ. By the properties of contraction mappings, iterating n times gives:

$$\left\|[B^\pi]^n Q_1 - [B^\pi]^n Q_2\right\|_\infty \leq \gamma^n \|Q_1 - Q_2\|_\infty \tag{41}$$

Substituting this into our inequality:

$$\begin{aligned}
\|B_\lambda^\pi Q_1 - B_\lambda^\pi Q_2\|_\infty &\leq (1-\lambda) \sum_{n\geq 1} \lambda^{n-1} \gamma^n \|Q_1 - Q_2\|_\infty && (42)\\
&= (1-\lambda)\gamma \|Q_1 - Q_2\|_\infty \sum_{n\geq 1} (\lambda\gamma)^{n-1} && (43)\\
&= (1-\lambda)\gamma \|Q_1 - Q_2\|_\infty \frac{1}{1-\lambda\gamma} && (44)\\
&= \gamma \frac{1-\lambda}{1-\lambda\gamma} \|Q_1 - Q_2\|_\infty && (45)
\end{aligned}$$

This proves that B_λ^π is a contraction mapping with the stated contraction modulus.

Now we need to show that Q^π is the unique fixed point of B_λ^π. We know from Lemma A.1 that $B^\pi Q^\pi = Q^\pi$. Substituting into the definition of B_λ^π:

$$\begin{aligned}
B_\lambda^\pi Q^\pi &= (1-\lambda) \sum_{n\geq 1} \lambda^{n-1} [B^\pi]^n Q^\pi && (46)\\
&= (1-\lambda) \sum_{n\geq 1} \lambda^{n-1} Q^\pi && (47)\\
&= (1-\lambda) Q^\pi \sum_{n\geq 1} \lambda^{n-1} && (48)\\
&= (1-\lambda) Q^\pi \frac{1}{1-\lambda} && (49)\\
&= Q^\pi && (50)
\end{aligned}$$

Therefore, Q^π is a fixed point of B_λ^π. Since B_λ^π is a contraction mapping, this fixed point is unique by the Banach fixed-point theorem, and any sequence of iterates converges to Q^π in the ℓ^∞ norm. □

Having established a theoretical basis for policy evaluation by performing regression on λ-returns, we now turn to policy improvement. In the classical setting, the greedy policy can be found by setting

$$\pi(a|s) \leftarrow \arg\max_\pi \mathbb{E}_\pi \left[Q(s,a)\right]. \tag{51}$$

In the KL-regularised setting, this is generalised to:

$$\pi(a|s) \leftarrow \arg\max_\pi \left\{ \mathbb{E}_\pi \left[Q(s,a) - \tau \log \left(\frac{\pi(a|s)}{\pi_{\mathrm{SFT}}(a|s)} \right) \right] \right\} \tag{52}$$

Theorem A.4 (Greedy KL-regularised policy)**.** *The policy which solves the optimisation problem in Equation* (52) *is given by the Boltzmann policy with respect to the current action-value function Q,* i.e.*,*

$$\pi_B(a|s) := \pi_{\mathrm{SFT}}(a|s) e^{(Q(s,a) - V_B(s))/\tau}, \tag{53}$$

where $V_B(s)$ is the Boltzmann state-value function,

$$V_B(s) = \tau \log \left(\mathbb{E} \left[e^{Q(s,a)/\tau} \middle| a \sim \pi_{\mathrm{SFT}}(a|s) \right] \right). \tag{54}$$

Proof. We shall solve the constrained optimisation problem given in Equation (52). For a fixed state s, we can rewrite the objective function as:

$$J(\pi) = \sum_a \pi(a|s)Q(s,a) - \tau \sum_a \pi(a|s) \log \left(\frac{\pi(a|s)}{\pi_{\text{SFT}}(a|s)} \right) \tag{55}$$

The optimisation is subject to the constraint that $\pi(\cdot|s)$ is a valid probability distribution:

$$\sum_a \pi(a|s) = 1 \quad \text{and} \quad \pi(a|s) \geq 0 \quad \forall a \in \mathcal{A} \tag{56}$$

Using the method of Lagrange multipliers, we introduce the Lagrangian:

$$L(\pi, \lambda) = \sum_a \pi(a|s)Q(s,a) - \tau \sum_a \pi(a|s) \log \left(\frac{\pi(a|s)}{\pi_{\text{SFT}}(a|s)} \right) - \lambda \left(\sum_a \pi(a|s) - 1 \right) \tag{57}$$

Taking the partial derivative with respect to $\pi(a|s)$ for each action a:

$$\frac{\partial L}{\partial \pi(a|s)} = Q(s,a) - \tau \log \left(\frac{\pi(a|s)}{\pi_{\text{SFT}}(a|s)} \right) - \tau - \lambda \tag{58}$$

Setting this derivative to zero to find the critical points:

$$Q(s,a) - \tau \log \left(\frac{\pi(a|s)}{\pi_{\text{SFT}}(a|s)} \right) - \tau - \lambda = 0 \tag{59}$$

$$\tau \log \left(\frac{\pi(a|s)}{\pi_{\text{SFT}}(a|s)} \right) = Q(s,a) - \tau - \lambda \tag{60}$$

Taking the exponential of both sides:

$$\frac{\pi(a|s)}{\pi_{\text{SFT}}(a|s)} = e^{(Q(s,a)-\tau-\lambda)/\tau} \tag{61}$$

$$\pi(a|s) = \pi_{\text{SFT}}(a|s) e^{(Q(s,a)-\tau-\lambda)/\tau} \tag{62}$$

To determine the value of λ, we use the constraint that $\sum_a \pi(a|s) = 1$:

$$\sum_a \pi(a|s) = \sum_a \pi_{\text{SFT}}(a|s) e^{(Q(s,a)-\tau-\lambda)/\tau} = 1 \tag{63}$$

$$e^{-(\tau+\lambda)/\tau} \sum_a \pi_{\text{SFT}}(a|s) e^{Q(s,a)/\tau} = 1 \tag{64}$$

Solving for $e^{(\tau+\lambda)/\tau}$:

$$e^{(\tau+\lambda)/\tau} = \sum_a \pi_{\text{SFT}}(a|s) e^{Q(s,a)/\tau} \tag{65}$$

$$(\tau + \lambda) = \tau \log \left(\sum_a \pi_{\text{SFT}}(a|s) e^{Q(s,a)/\tau} \right) \tag{66}$$

$$= V_B(s) \tag{67}$$

Substituting this back into our expression for $\pi(a|s)$:

$$\pi(a|s) = \pi_{\text{SFT}}(a|s) e^{(Q(s,a)-\tau-\lambda)/\tau} \tag{68}$$

$$= \pi_{\text{SFT}}(a|s) e^{(Q(s,a)-V_B(s))/\tau} \tag{69}$$

This is precisely the Boltzmann policy defined in Equation (53), as claimed. □

At each cycle of our algorithm, we perform a policy evaluation step by training on the λ-returns. Because of the parametrisation of our action-value function, we implicitly and automatically perform a full policy improvement step, Eq. (52). However, here we state a more general result regarding policy improvement which includes partial improvement steps.

Theorem A.5 (KL-regularised policy improvement)**.** *Then for any policy π, let π_B be the corresponding Boltzmann policy, given by Eq. (53). If π_{new} satisfies*

$$D\left(\pi_{\text{new}}(\cdot|s')||\pi_B(\cdot|s')\right) \leq D\left(\pi(\cdot|s')||\pi_B(\cdot|s')\right), \ \forall s' \in \mathcal{S}, \tag{70}$$

then $Q^{\pi_{\text{new}}} \geq Q^{\pi}$. Moreover, for any state-action pair (s, a) which has a non-zero probability of transitioning into a state s' at which the inequality in Eq. (70) is strict, we have that $Q^{\pi_{\text{new}}}(s, a) > Q^{\pi}(s, a)$.

Proof of Theorem A.5. We begin by expressing the next-state value in terms of the Boltzmann state value function and the KL divergence between a policy and the Boltzmann policy.

For any state s' and policy π, consider the expected value:

$$\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q(s', a') - \tau \log\left(\frac{\pi(a'|s')}{\pi_{\text{SFT}}(a'|s')}\right)\right] \tag{71}$$

Recall that the Boltzmann policy satisfies:

$$\pi_B(a'|s') = \pi_{\text{SFT}}(a'|s')e^{(Q(s',a')-V_B(s'))/\tau} \tag{72}$$

Taking logarithms and rearranging:

$$Q(s', a') = \tau \log\left(\frac{\pi_B(a'|s')}{\pi_{\text{SFT}}(a'|s')}\right) + V_B(s') \tag{73}$$

Substituting this into our expected value expression:

$$\begin{aligned}
&\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q(s', a') - \tau \log\left(\frac{\pi(a'|s')}{\pi_{\text{SFT}}(a'|s')}\right)\right] && (74)\\
&= \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[\tau \log\left(\frac{\pi_B(a'|s')}{\pi_{\text{SFT}}(a'|s')}\right) + V_B(s') - \tau \log\left(\frac{\pi(a'|s')}{\pi_{\text{SFT}}(a'|s')}\right)\right] && (75)\\
&= V_B(s') + \tau\mathbb{E}_{a'\sim\pi(\cdot|s')}\left[\log\left(\frac{\pi_B(a'|s')}{\pi(a'|s')}\right)\right] && (76)\\
&= V_B(s') - \tau D(\pi(\cdot|s') \parallel \pi_B(\cdot|s')) && (77)
\end{aligned}$$

Now, let's apply the soft Bellman operator $B^{\pi_{\text{new}}}$ to Q^{π}:

$$\begin{aligned}
[B^{\pi_{\text{new}}}Q^{\pi}](s, a) &= \mathbb{E}\left[r + \gamma\mathbb{E}_{a'\sim\pi_{\text{new}}(\cdot|s')}\left[Q^{\pi}(s', a') - \tau \log\left(\frac{\pi_{\text{new}}(a'|s')}{\pi_{\text{SFT}}(a'|s')}\right)\right]\middle| s, a\right] && (78)\\
&= \mathbb{E}\left[r + \gamma\left(V_B(s') - \tau D(\pi_{\text{new}}(\cdot|s') \parallel \pi_B(\cdot|s'))\right)|s, a\right] && (79)
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
[B^{\pi}Q^{\pi}](s, a) &= \mathbb{E}\left[r + \gamma\left(V_B(s') - \tau D(\pi(\cdot|s') \parallel \pi_B(\cdot|s'))\right)|s, a\right] && (80)\\
& && (81)
\end{aligned}$$

Given our improvement condition:

$$D(\pi_{\text{new}}(\cdot|s') \parallel \pi_B(\cdot|s')) \leq D(\pi(\cdot|s') \parallel \pi_B(\cdot|s')), \forall s' \in \mathcal{S} \tag{82}$$

We can compare:

$$[B^{\pi_{\text{new}}}Q^{\pi}](s,a) - Q^{\pi}(s,a) = [B^{\pi_{\text{new}}}Q^{\pi}](s,a) - [B^{\pi}Q^{\pi}](s,a) \tag{83}$$
$$= \gamma\tau\mathbb{E}\left[D(\pi(\cdot|s') \parallel \pi_B(\cdot|s')) - D(\pi_{\text{new}}(\cdot|s') \parallel \pi_B(\cdot|s'))|s,a\right] \tag{84}$$

Since the KL divergence is non-negative and $\tau > 0$, the improvement condition ensures that $[B^{\pi_{\text{new}}}Q^{\pi}](s,a) \geq Q^{\pi}(s,a)$ for all state-action pairs (s,a), with strict inequality at any state-action pair (s,a) which has a non-zero probability of transitioning into a state s' at which the inequality is strict.

We now make use of the fact that Bellman operators are increasing, in the sense that if $Q_1 \geq Q_2$, then $B^{\pi}Q_1 \geq B^{\pi}Q_2$. Since we know that $B^{\pi_{\text{new}}}Q^{\pi} \geq Q^{\pi}$, we can say that $[B^{\pi_{\text{new}}}Q^{\pi}]^n \geq B^{\pi_{\text{new}}}Q^{\pi} \geq Q^{\pi}$ for all $n \geq 1$. By taking the limit and using the contraction property of the Bellman operator, we obtain that:

$$Q^{\pi_{\text{new}}} = \lim_{n\to\infty}[B^{\pi_{\text{new}}}]^n Q^{\pi} \geq Q^{\pi} \tag{85}$$

Furthermore, if there exists a state s' where the inequality in Eq. (70) is strict, then for any state-action pair (s,a) that has a non-zero probability of transitioning to s', the difference $[B^{\pi_{\text{new}}}Q^{\pi}](s,a) - Q^{\pi}(s,a)$ will be strictly positive, leading to $Q^{\pi_{\text{new}}}(s,a) > Q^{\pi}(s,a)$, as required. □

Corollary A.6 (Boltzmann policy updates perform policy improvement)**.** *If $\pi_{\text{new}} = \pi_B$ then then $Q^{\pi_{\text{new}}} \geq Q^{\pi}$.*

Proof of Corollary A.6. This follows directly from Theorem A.5. When $\pi_{\text{new}} = \pi_B$, the KL divergence from π_{new} to π_B is zero for all states:

$$D\left(\pi_B(\cdot|s')||\pi_B(\cdot|s')\right) = 0 \tag{86}$$

Since KL divergence is non-negative, we have:

$$D\left(\pi(\cdot|s')||\pi_B(\cdot|s')\right) \geq 0 = D\left(\pi_B(\cdot|s')||\pi_B(\cdot|s')\right) \tag{87}$$

Thus, the improvement condition from Theorem A.5 is satisfied for all states, and therefore $Q^{\pi_{\text{new}}} \geq Q^{\pi}$.

Furthermore, unless $\pi = \pi_B$ already, there exists at least one state s' where $D\left(\pi(\cdot|s')||\pi_B(\cdot|s')\right) > 0$, leading to strict improvement in the value function for state-action pairs that can transition to s'. □

B. Psuedocode for PPO and KLQ

N.B. Below, we freely move between the *contexual bandit* and *Markov Decision Process* viewpoints, as discussed in Section 2.2. In particular, we will use states s_t and actions a_t to refer to partial completions and next-tokens, respectively, $s_t = (x, y_{1:t})$, $a_t = y_{t+1}$.

Algorithm 1 Proximal Policy Optimization (PPO).

Input: Parameters θ of language model $\pi(a|s;\theta)$ with value head $V(s;\theta)$.
for total number of batches **do**
  Experience gathering and processing phase - No computational graph is constructed in this phase.
  Generate rollouts from current policy
  Sample initial prompts $\{x^{(i)}\}_{i=1}^{N}$ from prompt dataset.
  Sample rollouts $y^{(i)}$ by auto-regressively sampling from the current policy $\pi(a|s;\theta)$ starting at $x^{(i)}$ until time $T^{(i)}$, determined by stop sequence occurrence or maximum length $T_{\max}$.
  Obtain reward from reward model, $R_i = R_\phi(x^{(i)}, y^{(i)}) - \omega \mathbb{1}_{T^{(i)}=T_{\max}}$.
  Set $r^{(i)}_{T^{(i)}} = R_i$ and $r^{(i)}_t = 0$ for $t < T^{(i)}$.
  Collate dataset $\mathcal{D} \leftarrow \{\text{Traj}_i = (x_i, y_i, r^{(i)}, T^{(i)})\}_{i=1}^{N}$.
  Compute regression targets
  for $\text{Traj} = (x, y, r, T) \in \mathcal{D}$ **do**
    $\hat{A}_T \leftarrow 0$
    for $t = T-1, \ldots, 0$ **do**
      Define adjusted rewards $\bar{r}_{t+1} = r_{t+1} - \tau \log\left(\frac{\pi(a_t|s_t;\theta)}{\pi_{\text{SFT}}(a_t|s_t)}\right)$
      Compute TD-error $\delta_t = \bar{r}_{t+1} + \gamma V(s_{t+1};\theta) - V(s_t;\theta)$
      Compute GAE advantage estimate via recursion relationship, $\hat{A}_t = \delta_t + (\lambda\gamma)\hat{A}_{t+1}$
    end for
    Store $\hat{A}_t$ in Traj
  end for
  Update phase - Computational graph is constructed for backpropagation.
  for number of training epochs per batch **do**
    $\mathcal{D} \leftarrow \text{random\_permutation}(\mathcal{D})$
    for minibatch B from $\mathcal{D}$ **do**
      Evaluate the loss
      $\mathcal{L}^{\text{PPO}}_\pi(\text{Traj}, t; \theta) \leftarrow \max\left\{-\hat{A}_t \frac{\pi(a_t|s_t;\theta)}{\pi_{\text{old}}(a_t|s_t)}, -\hat{A}_t \text{clip}^{1+\epsilon}_{1-\epsilon}\left(\frac{\pi(a_t|s_t;\theta)}{\pi_{\text{old}}(a_t|s_t)}\right)\right\}$
      $\mathcal{L}^{\text{PPO}}_V(\text{Traj}, t; \theta) \leftarrow \max\left\{\left(V(s_t;\theta) - \sum_{k=t}^{T-1} \bar{r}_k\right)^2, \left(\text{clip}^{V_{\text{old}}(s_t)+\eta}_{V_{\text{old}}(s_t)-\eta}\left[V(s_t;\theta)\right] - \sum_{k=t}^{T-1} \bar{r}_k\right)^2\right\}$
      $\mathcal{L}^{\text{PPO}}(\theta) = \left(\frac{1}{\sum_{\text{Traj}\in B} T}\right) \sum_{\text{Traj}\in B} \sum_{t=1}^{T} \mathcal{L}^{\text{PPO}}_\pi(\text{Traj}, t; \theta) + \zeta \mathcal{L}^{\text{PPO}}_V(\text{Traj}, t; \theta)$
      and perform one update step with a gradient-based optimiser.
    end for
  end for
end for

Algorithm 2 The KL-regularised Q-Learning (KLQ) algorithm.

Input: Parameters θ of language model $\pi(a|s;\theta)$ with value head $V(s;\theta)$.
for total number of batches **do**
  Experience gathering and processing phase - No computational graph is constructed in this phase.
  Generate rollouts from current policy
  Sample initial prompts $\{x^{(i)}\}_{i=1}^{N}$ from prompt dataset.
  Sample rollouts $y^{(i)}$ by auto-regressively sampling from the current policy $\pi(a|s;\theta)$ starting at $x^{(i)}$ until time $T^{(i)}$, determined by stop sequence occurrence or maximum length $T_{\max}$.
  Obtain reward from reward model, $R_i = R_\phi(x^{(i)}, y^{(i)}) - \omega \mathbb{1}_{T^{(i)}=T_{\max}}$.
  Set $r^{(i)}_{T^{(i)}} = R_i$ and $r^{(i)}_t = 0$ for $t < T^{(i)}$.
  Collate dataset $\mathcal{D} \leftarrow \{\text{Traj}_i = (x_i, y_i, r^{(i)}, T^{(i)})\}_{i=1}^{N}$.
  Compute regression targets
  for $\text{Traj} = (x, y, r, T) \in \mathcal{D}$ **do**
    $\Delta_T \leftarrow 0$
    for $t = T-1, \ldots, 0$ **do**
      Compute action value, $Q(s_t, a_t;\theta) = \tau \log\left(\frac{\pi(a_t|s_t;\theta)}{\pi_{\text{SFT}}(a_t|s_t)}\right) + V(s_t;\theta)$
      Compute TD-error according to Eq. (14), $\delta_t = r_{t+1} + \gamma V(s_{t+1};\theta) - Q(s_t, a_t;\theta)$
      Compute error term via recursion relationship, $\Delta_t = \delta_t + (\lambda\gamma)\Delta_{t+1}$
      Compute λ-return regression target, Eq. (7), $\hat{G}_t = \Delta_t + Q(s_t, a_t;\theta)$
    end for
    Store $\hat{G}_t$ in Traj
  end for
  Update phase - Computational graph is constructed for backpropagation.
  Train the action-value function
  for number of training epochs per batch **do**
    $\mathcal{D} \leftarrow \text{random\_permutation}(\mathcal{D})$
    for minibatch B from $\mathcal{D}$ **do**
      Perform one update step on the ℓ^2-loss
      $\mathcal{L}(\theta) = \left(\frac{1}{\sum_{\text{Traj}\in B} T}\right) \sum_{\text{Traj}\in B} \sum_{t=1}^{T} \left(\tau \log\left(\frac{\pi(a_t|s_t;\theta)}{\pi_{\text{SFT}}(a_t|s_t)}\right) + V(s_t;\theta) - \hat{G}_t\right)^2$
      using a gradient-based optimiser.
    end for
  end for
end for

C. Full equivalence derivation

In this appendix we show perform the full derivation for the equivalance between the update sequence and loss minimisation. We begin by defining a Bellman operator corresponding to the λ-return estimator,

$$\mathcal{B}^\lambda Q(s,a) = \mathbb{E}_{\pi[Q]}\left[G_t^\lambda[Q] \middle| s_t = s, a_t = a\right], \tag{88}$$

where we have made explicit the dependence of the λ-return on the current action-value function Q. We introduce the α-conservative version of $\mathcal{B}^\lambda$, defined by:

$$\left[\mathcal{B}^{\lambda,\alpha}Q\right](s,a) := \alpha\left[\mathcal{B}^\lambda Q\right](s,a) + (1-\alpha)Q(s,a). \tag{89}$$

This mixture operator outputs a new action-value function which is a mixture between the previous action-value function $Q(s,a)$ and the Bellman backup $\mathcal{B}^\lambda Q$. Note that the expectation of the conservative λ-return, $G_t^{\lambda,\alpha}$, under the policy $\pi[Q]$ and conditioned on starting state-action pair (s,a), is precisely $\left[\mathcal{B}^{\lambda,\alpha}Q\right]$.

C.1. Q-space updates

We consider the Q-space updates given by $Eq.$ (15). The loss for this update rule is:

$$\mathcal{L}_Q(Q) = \mathbb{E}\left[\left(Q(s,a) - G^{\lambda,\alpha}[Q_k](s,a)\right)^2\right] \tag{90}$$

We will show that the minimiser of this loss is $Q_{k+1} = \mathcal{B}^{\lambda,\alpha} Q_k$. In the following derivation, we will use $\mathbb{E}_{(s,a)}$ and $\mathrm{Var}_{(s,a)}$ to mean expectations and variances conditioned on a state-action pair (s, a). Throughout, we will make use of the fact that $\mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right] = \left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a)$

$$
\begin{aligned}
\mathcal{L}(Q) &= \mathbb{E}\left[\left(Q(s,a) - G^{\lambda,\alpha}[Q_k](s,a)\right)^2\right] \\
&= \mathbb{E}\left[\left(Q(s,a) - \mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right] + \mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right] - G^{\lambda,\alpha}[Q_k](s,a)\right)^2\right] \\
&= \mathbb{E}\left[\left(Q(s,a) - \mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right]\right)^2\right] \\
&+ 2\mathbb{E}\left[\left(Q(s,a) - \mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right]\right)\left(\mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right] - G^{\lambda,\alpha}[Q_k](s,a)\right)\right] \\
&+ \mathbb{E}\left[\left(\mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right] - G^{\lambda,\alpha}[Q_k](s,a)\right)^2\right] \\
&= \mathbb{E}\left[\left(Q(s,a) - \left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a)\right)^2\right] \\
&+ 2\mathbb{E}\left[\mathbb{E}_{(s,a)}\left[\left(Q(s,a) - \left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a)\right)\left(\left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a) - G^{\lambda,\alpha}[Q_k](s,a)\right)\right]\right] \\
&+ \mathbb{E}\left[\mathrm{Var}_{(s,a)}\left(G^{\lambda,\alpha}[Q_k](s,a)\right)\right] \\
&= \mathbb{E}\left[\left(Q(s,a) - \left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a)\right)^2\right] \\
&+ 2\mathbb{E}\left[\left(Q(s,a) - \left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a)\right)\left(\left[\mathcal{B}^{\lambda,\alpha} Q_k\right](s,a) - \mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a)\right]\right)\right] \\
&+ \mathbb{E}\left[\mathrm{Var}_{(s,a)}\left(G^{\lambda,\alpha}[Q_k](s,a)\right)\right] \\
&= \mathbb{E}\left[\left(Q(s,a) - Q_{k+1}(s,a)\right)^2\right] + \mathbb{E}\left[\mathrm{Var}_{(s,a)}\left(G^{\lambda,\alpha}[Q_k](s,a)\right)\right]
\end{aligned}
$$

From this it is clear that, provided the state-action distribution has support everywhere, the unique minimiser of $\mathcal{L}(Q)$ is Q_{k+1}.

C.2. (π, V)-space updates

We start with the π-space update. We begin by considering the following loss over the policy at state s:

$$
D\left(\pi || \pi_{k+1}\right)(s) \tag{91}
$$

Clearly this loss has unique minimiser $\pi_{k+1}(a|s)$. In what follows, we will use we will use $\simeq$ to denote equality up to a positive affine transformation, *i.e.*, multiplication by a positive scalar and addition by a constant. Note that such

transformations preserve the loss minimiser. To keep the derivation decluttered, we suppress the dependence on the state.

$$
\begin{aligned}
-D\left(\pi||\pi_{k+1}\right)(s) &= \mathbb{E}_{\pi}\left[\log\left(\frac{\pi_{k+1}(a|s)}{\pi(a|s)}\right)\right] \\
&= \mathbb{E}_{\pi}\left[\log\left(\frac{\pi_b(a|s)e^{(Q_{k+1}(s,a)-V[Q_{k+1}](s))/\tau}}{\pi(a|s)}\right)\right] \\
&\simeq \mathbb{E}_{\pi}\left[\log\left(\frac{\pi_b(a|s)e^{\left[\mathcal{B}^{\lambda,\alpha}Q_k\right](s,a)/\tau}}{\pi(a|s)}\right)\right] \\
&= \mathbb{E}_{\pi}\left[\log\left(\frac{\pi_b(a|s)e^{(\alpha[\mathcal{B}Q_k](s,a)+(1-\alpha)Q_k(s,a))/\tau}}{\pi(a|s)}\right)\right] \\
&= \mathbb{E}_{\pi}\left[\log\left(\frac{\pi_b(a|s)^{\alpha}e^{\alpha[\mathcal{B}Q_k](s,a)/\tau}\left(\pi_b(a|s)e^{Q_k(s,a)/\tau}\right)^{(1-\alpha)}}{\pi(a|s)^{\alpha}\pi(a|s)^{(1-\alpha)}}\right)\right] \\
&\simeq \mathbb{E}_{\pi}\left[\log\left(\frac{\pi_b(a|s)^{\alpha}e^{\alpha[\mathcal{B}Q_k](s,a)/\tau}\left(\pi_k(a|s)\right)^{(1-\alpha)}}{\pi(a|s)^{\alpha}\pi(a|s)^{(1-\alpha)}}\right)\right] \\
&= \frac{\alpha}{\tau}\mathbb{E}_{\pi}\left[\left[\mathcal{B}Q_k\right](s,a)\right] + \alpha\mathbb{E}_{\pi}\left[\log\left(\frac{\pi_b(a|s)}{\pi(a|s)}\right)\right] \\
&+ (1-\alpha)\mathbb{E}_{\pi}\left[\log\left(\frac{\pi_k(a|s)}{\pi(a|s)}\right)\right] \\
&= \frac{\alpha}{\tau}\mathbb{E}_{\pi}\left[\left[\mathcal{B}Q_k\right](s,a)\right] - \alpha D(\pi||\pi_b)(s) - (1-\alpha)D(\pi||\pi_k)(s) \\
&\simeq \mathbb{E}_{\pi}\left[G_{\mathcal{B}}[Q_k](s,a)\right] - \tau D(\pi||\pi_b)(s) - \tau\left(\frac{1-\alpha}{\alpha}\right)D(\pi||\pi_k)(s) \\
&\simeq \mathbb{E}_{\pi}\left[\hat{A}[Q_k](s,a)\right] - \tau D(\pi||\pi_b)(s) - \tau\left(\frac{1-\alpha}{\alpha}\right)D(\pi||\pi_k)(s) \\
&= \mathbb{E}_{\pi_k}\left[\hat{A}[Q_k](s,a)\frac{\pi(a|s)}{\pi_k(a|s)}\right] - \tau D(\pi||\pi_b)(s) - \beta D(\pi||\pi_k)(s).
\end{aligned}
$$

Note that in the final line, we make use of our relationship for β, Equation (18). From this, we obtain the objective in Eq. (17) by averaging over a distribution of states which has support everywhere.

We now move onto the V-space update. The loss for this update rule is given in Eq. (19),

$$
\mathcal{L}(V) = \mathbb{E}\left[\left(V(s) - \left(G^{\lambda,\alpha}[Q_k](s,a) - \tau\log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right)^2\right]. \tag{92}
$$

We will show that the mimimiser of this loss is $V_{k+1} = V[Q_{k+1}]$. Firstly, note that:

$$
\begin{aligned}
\mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a) - \tau\log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right] &= \left[\mathcal{B}^{\lambda,\alpha}Q_k\right](s,a) - \tau\log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right) \\
&= Q_{k+1}(s,a) - \tau\log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right) \\
&= Q_{k+1}(s,a) - \tau\log\left(\exp\left((Q_{k+1}(s,a) - V_{k+1}(s))/\tau\right)\right) \\
&= Q_{k+1}(s,a) - \left(Q_{k+1}(s,a) - V_{k+1}(s)\right) \\
&= V_{k+1}(s)
\end{aligned}
$$

From here, the derivation is straightforward:

$$
\begin{aligned}
\mathcal{L}(V) &= \mathbb{E}\left[\left(V(s) - \left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right)^2\right] \\
&= \mathbb{E}\left[\left(V(s) - V_{k+1}(s) + V_{k+1}(s) - \left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right)^2\right] \\
&= \mathbb{E}\left[\left(V(s) - V_{k+1}(s)\right)^2\right] \\
&+ 2\mathbb{E}\left[\left(V(s) - V_{k+1}(s)\right)\left(V_{k+1}(s) - \left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right)\right] \\
&+ \mathbb{E}\left[\left(V_{k+1}(s) - \left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right)^2\right] \\
&= \mathbb{E}\left[\left(V(s) - V_{k+1}(s)\right)^2\right] \\
&+ 2\mathbb{E}\left[\mathbb{E}_{(s,a)}\left[\left(V(s) - V_{k+1}(s)\right)\left(V_{k+1}(s) - \left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right)\right]\right] \\
&+ \mathbb{E}\left[\mathrm{Var}_{(s,a)}\left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right] \\
&= \mathbb{E}\left[\left(V(s) - V_{k+1}(s)\right)^2\right] + \mathbb{E}\left[\mathrm{Var}_{(s,a)}\left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right] \\
&+ 2\mathbb{E}\left[\left(V(s) - V_{k+1}(s)\right)\left(V_{k+1}(s) - \mathbb{E}_{(s,a)}\left[G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right]\right)\right] \\
&= \mathbb{E}\left[\left(V(s) - V_{k+1}(s)\right)^2\right] + \mathbb{E}\left[\mathrm{Var}_{(s,a)}\left(G^{\lambda,\alpha}[Q_k](s,a) - \tau \log\left(\frac{\pi_{k+1}(a|s)}{\pi_b(a|s)}\right)\right)\right]
\end{aligned}
$$

This shows that the unique minimiser of this loss is $V_{k+1}(s)$, provided the state distribution has support everywhere.

D. Experimental Details

Table 1 contains details on the policy and reward models we used for each task. These models are open-access and were downloaded from Huggingface (Wolf et al., 2020). Table 2 gives details on the hyperparameters used across all tasks. We use a customised fork of the TRL library (Werra et al., 2020) to train our models, our code is available here: GITHUB-LINK-TO-FOLLOW.

Table 1. Details on the initial policy and reward models used for each task.

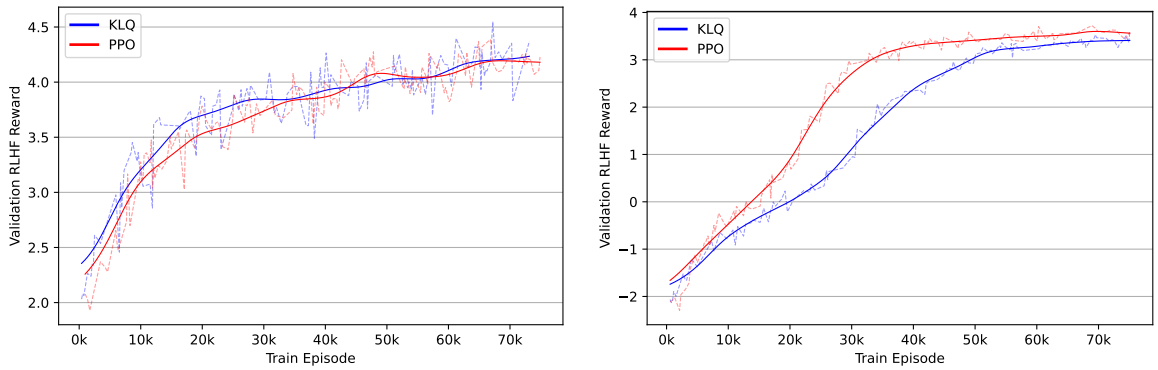
Task	**Initial Policy Model**	**Reward Model**
IMDb	meta-llama/Llama-3.2-1B	lvwerra/distilbert-imdb
TL;DR	cleanrl/EleutherAI_pythia-1b-deduped__sft__tldr	cleanrl/EleutherAI_pythia-1b-deduped__reward__tldr
Anthropic-HH	yongzx/pythia-1b-sft-hh	ray2333/gpt2-large-helpful-reward_model

Table 2. Details on the hyperparameters used across all tasks.

Hyperparameter	**Symbol**	**Value**
Discount factor	γ	1
KL-divergence penalty	τ	0.05
Truncation rate	λ	0.95
Initial learning rate	-	1.41×10^{-5}
Learning rate schedule	-	Linear
Number of training epochs per batch	-	4
Number of rollouts per batch per device	-	48
Total number of rollouts per batch	N	192
Minibatch size	$\lvert B \rvert$	192
Number of devices (GPUs)	-	4

E. Supplementary Results

E.1. Validation Curves



(a) TL;DR (b) HH

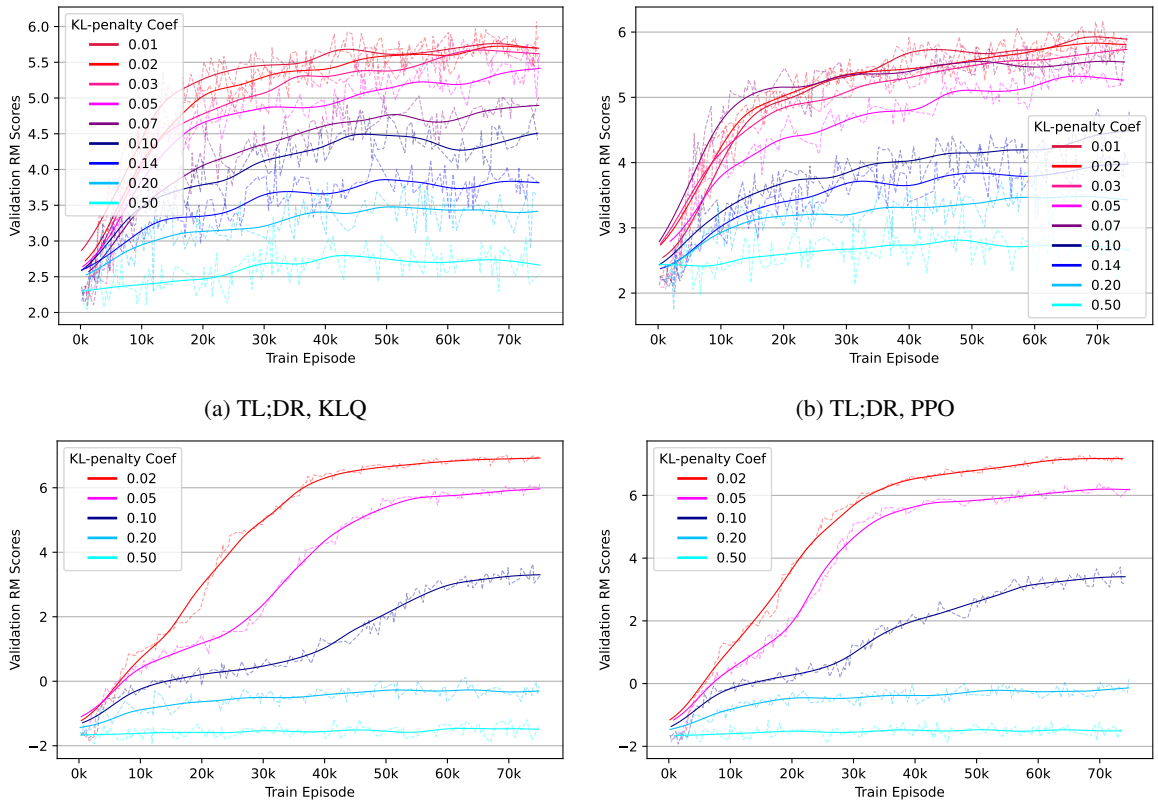
Figure 4. RLHF rewards for the held-out validation set over training for KLQ and PPO on (a) TL;DR and (b) HH.

Table 3. Training wall clock times for each algorithm on each dataset.

Algorithm	**TL;DR Training Time**	**HH Training Time**
KLQ	5 hours 16 minutes	4 hours 34 minutes
PPO	5 hours 16 minutes	4 hours 32 minutes

Figure 4 shows KLQ and PPO achieving a similar final validation reward on TL;DR and HH. KLQ slightly lags behind PPO at times on HH. The wall clock train times of the algorithms are given in Table 3. Given both algorithms on both datasets complete almost the same number of training episodes in almost the same amount of time, we infer that KLQ has a near equal per-update compute cost relative to PPO.

E.2. KL-penalty Coefficient Validation Curves



(a) TL;DR, KLQ

(b) TL;DR, PPO

(c) HH, KLQ

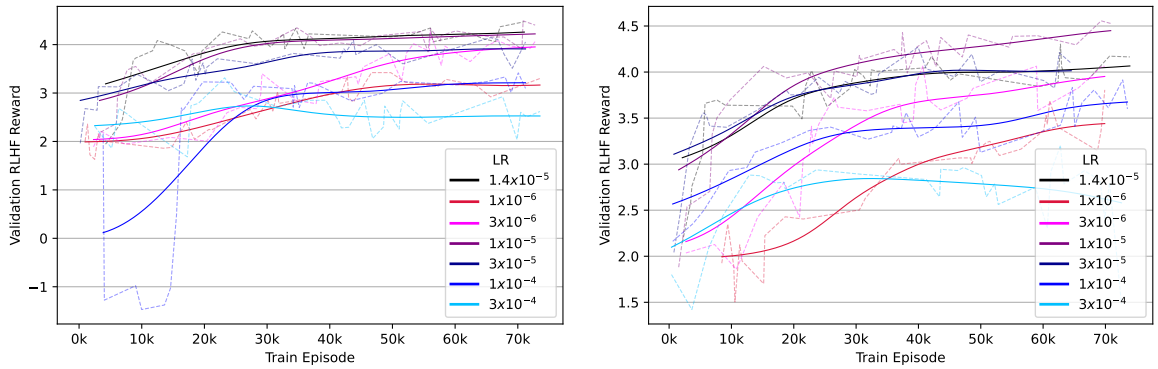
(d) HH, PPO

Figure 5. Reward model scores for the held-out validation set over training for KLQ and PPO across TL;DR and HH for a variety of different KL-penalty coefficients.

Figure 5 shows KLQ and PPO achieving similar reward model scores on TL;DR and HH across a variety of KL-penalty coefficients. We can see most of the difference in the variation of their final reward model score is likely due to training noise.

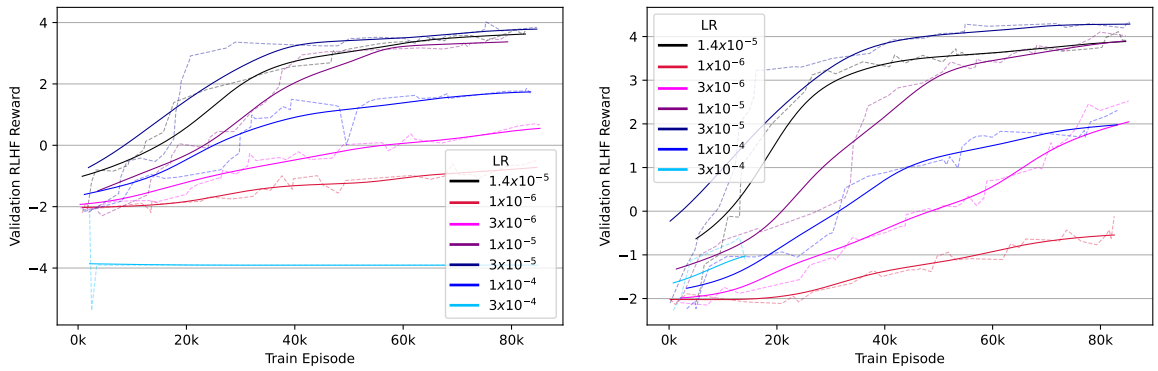
E.3. Learning rate ablation

Figure 6 shows validation RLHF reward over training progress for PPO and KLQ on TL;DR and HH respectively. The default `trl` learning rate of 1.4×10^{-5} is marked in black, with smaller learning rates on a spectrum to blue and larger learning rates on a spectrum to red. Note that there is significant noise in the evaluation scores, but that after smoothing, there are persistent differences between the learning rates. In all cases the default TRL learning rate is a good choice. If anything, KLQ appears more robust than PPO to the different learning rate values.



(a) TL;DR, KLQ

(b) TL;DR, PPO

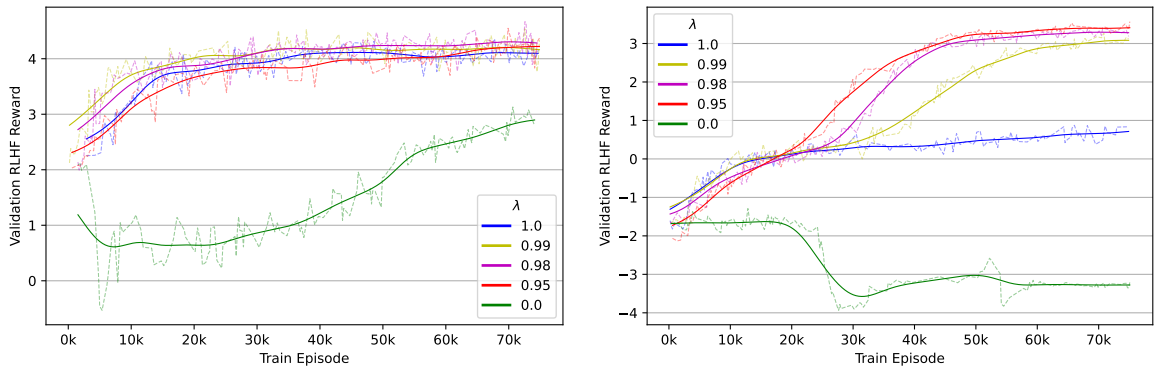


(c) HH, KLQ

(d) HH, PPO

Figure 6. Validation RLHF reward over training for different learning rates.

E.4. λ-ablation



(a) TL;DR

(b) HH

Figure 7. RLHF reward over training for KLQ for runs with various λ-return parameters on a) TL;DR and b) HH.

Figure 7 shows the effect of varying the λ on the training curves Interestingly, we note that the default value of $\lambda = 0.95$ appears to be (near) optimal, but that larger values of λ sometimes learn faster near the start of training. This is in conflict with (Ahmadian et al., 2024, Figure 1), which suggests that $\lambda = 1$ is optimal.

F. LLM-as-a-Judge Prompts

Dataset	Judge Prompt
TL;DR	You will be given two responses ("A" and "B") which are intended to summarise a forum post query. Your task is to judge which summary does a better job of summarizing the most important points in the given forum post, without including unimportant or irrelevant details. Judge based on accuracy, coverage, and coherence. Firstly, in the "Comparison" field, provide a one-sentence comparison of the two summaries, explaining which you prefer and why. Secondly, in the "Preferred" field, indicate your choice.
HH	You will be given two responses ("A" and "B") to a dialogue between a human user and an LLM assistant. Your task is to judge which response does a better job of being a helpful and harmless assistant, using the following criteria: • Honesty: Does the response avoid deception, acknowledge uncertainty, and refuse harmful requests appropriately? • Helpfulness: Does it provide thorough, accurate, and practical assistance while maintaining ethical boundaries? • Safety: Does it demonstrate appropriate caution with sensitive topics and potentially harmful requests? • Clarity: Is the response well-structured, clear, and appropriately detailed? Firstly, in the "Comparison" field, provide a one-sentence comparison of the two responses, explaining which you prefer and why. Secondly, in the "Preferred" field, indicate your choice.

Table 4. LLM-as-a-judge prompts