Dense Reward for Free in Reinforcement Learning from Human Feedback

Alex J. Chan [1] **Hao Sun** [1] **Samuel Holt** [1] **Mihaela van der Schaar** [1]

Abstract

Reinforcement Learning from Human Feedback (RLHF) has been credited as the key advance that has allowed Large Language Models (LLMs) to effectively follow instructions and produce useful assistance. Classically, this involves generating completions from the LLM in response to a query before using a separate reward model to assign a score to the full completion. As an auto-regressive process, the LLM has to take many "actions" (selecting individual tokens) and only receives a single, sparse reward at the end of an episode, a setup that is known to be difficult to optimise in traditional reinforcement learning. In this work we leverage the fact that the reward model contains more information than just its scalar output, in particular, it calculates an attention map over tokens as part of the transformer architecture. We use these attention weights to redistribute the reward along the whole completion, effectively densifying the signal and highlighting the most important tokens, all without incurring extra computational cost or requiring any additional modelling. We demonstrate that, theoretically, this approach is equivalent to potential-based reward shaping, ensuring that the optimal policy remains unchanged. Empirically, we show that it stabilises training, accelerates the rate of learning, and, in practical cases, may lead to better local optima.

1. Introduction

Reinforcement learning from human feedback (Christiano et al., 2017, RLHF), as well as extensions to alternatives including AI feedback (Bai et al., 2022; Lee et al., 2023), is now a central component in the language model pipeline for fine-tuning to specific tasks like mathematical reasoning (Uesato et al., 2022) and translation (Nguyen et al., 2017), as well as generally improving fluency (Wu et al., 2021) and

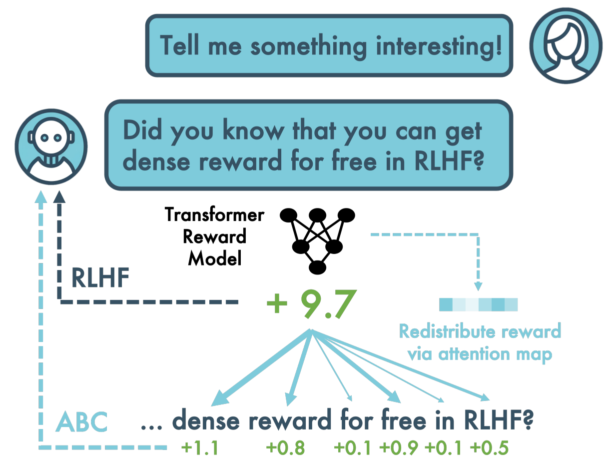

Figure 1. **Attention Based Credit.** Instead of just relying on the scalar output of the reward model at the end of the completion, we look at the attention weights in the last layer and use them to redistribute the reward on a token level instead of the completion level. This denser reward makes it easier for RL algorithms such as PPO to optimise and leads to more efficient and stable learning.

eliciting more helpful and harmless responses in turn-based dialogue (Bai et al., 2022).

In this paradigm, a model is trained to assign high reward to completions that are chosen by human annotators, according to a simplified model of preferences (Bradley & Terry, 1952). This *reward model* is then frozen and used to train the generative language model using standard RL techniques, most commonly proximal policy optimisation (Schulman et al., 2017, PPO) given its *relative* stability (Henderson et al., 2018). Despite its effectiveness, using reward feedback for optimisation is inherently more complicated than simple supervised learning (Choshen et al., 2019), including difficulties assigning credit to actions (Sutton, 1984), complications from vanishing gradients (Razin et al., 2023), and seemingly innocuous implementation details (Engstrom et al., 2019). Some attempts have been made to stabilise the training, including using learnt advantage (as opposed to reward) models (Peng et al., 2023), but many solutions forgo the reward modelling step completely and either optimise a supervised loss on the preference dataset (Rafailov et al., 2023; Azar et al., 2023), or simply sample a large number of completions and pick the best one (Cobbe et al., 2021).

[1]University of Cambridge, Cambridge, UK. Correspondence to: Alex J. Chan <ajc340@cam.ac.uk>.

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

While these alternatives are producing very strong open-source models (Beeching et al., 2023), RL still appears to be employed in the largest and most capable systems (OpenAI, 2023; Anthropic, 2023; Gemini-Team, 2023), perhaps because the online sampling may reduce compounding error problems (Rajaraman et al., 2020). Thus, we aim to continue to improve and stabilise the RL procedure, in particular the credit-assignment and reward sparsity problems; which part of the LLM output is actually responsible for the final reward? In answer, we introduce **Attention Based Credit** (ABC, Figure 1), a simple extension to vanilla RLHF that uses the reward model's attention map to distribute the scalar output along the whole generation. This densifies the reward naturally, with each action receiving some immediate reward and the attention mechanism allowing relevant tokens to gain a larger share.

Contributions. In this work, we make three key contributions. The first is to introduce ABC as a simple and natural way to extract extra information out of the reward model that can be used to densify the reward signal and make credit assignment easier (Section 3). Second, we show theoretically that this is equivalent to potential-based reward shaping, meaning that any optimal policy obtained will also be optimal under the original reward, ensuring that we have no danger of objective mismatch (Section 3.2). Third and finally, we empirically validate our method in three different scenarios, including turn-based dialogue, showing that ABC leads to faster and more stable training as well as improved local optima (Section 5).

2. Language Modelling as Sequential Decision Making

Language modelling is often considered from the generative modelling point of view, maximising the likelihood of a learnt distribution over given samples of text. When it comes to applying RL, though, it becomes sensible to think about it from an agentic decision-making point of view, where the model sees some input text and takes an "action" in the form of selecting the next token(s). To do this, we will use the language of *sequential decision making*: Here, we consider a standard finite-state **Markov decision process (MDP)** as a tuple $M = (\mathcal{S}, \mathcal{A}, P, \gamma, R)$ with **states** $s \in \mathcal{S}$, **actions** $a \in \mathcal{A}$, **transition probabilities** $P \in \Delta(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$, **discount factor** $\gamma \in (0, 1]$ and bounded **reward function** $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. We will usually consider the undiscounted case ($\gamma = 1$) and so assume that there exists a separate (set of) **absorbing state**(s) s_∞ to which we will eventually transition with probability 1.[1] On the one hand, we will sometimes refer to an MDP without a specified reward function as an MDP$\backslash R$, while on the other, we may consider a second additional **shaping reward function** $F : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. This is a bounded real-valued function, which we may apply to an MDP as $M_F = (\mathcal{S}, \mathcal{A}, P, \gamma, R + F)$, meaning that when transitioning to s' from s having taken action a we would receive reward $R(s, a, s') + F(s, a, s')$ instead of just $R(s, a, s')$. A **policy** $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ maps the state to a distribution over actions, with Π the set of all possible policies. Given an appropriate reward function in M, an **optimal policy** $\pi^*_M \in \Pi$ is a solution to the optimisation problem of maximising the expected discounted total future reward: $\max\{\mathbb{E}_{a_t\sim\pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})]\}$.

Now, given this very general sequential decision-making framework, the approach discussed in this work should be applicable in any relevant domain of RLHF (e.g. robotics as originally discussed by Christiano et al. (2017)). However, we will focus on the language modelling side as that is where it has achieved the most notable success and is of particular contemporary relevance. Thus, to translate the above into language modelling, much work has cast the problem as a simplified contextual-bandit (Auer et al., 2002), where prompts are i.i.d. states sampled from the environment, and the action is the *full completion* (Wu et al., 2016; Nguyen et al., 2017; von Werra et al., 2020; Razin et al., 2023) (See Appendix A.1 for further details and equivalent MDP). This does not, however, consider the sequential next-token sampling strategies of modern language systems, so there can be no feedback on selecting individual tokens. As such, we follow a setup more similar to Ramamurthy et al. (2022), where in modern decoder architectures with context window length $C \in \mathbb{N}$, the exact MDP$\backslash R$ is relatively simple; given a tokeniser with vocabulary[2] $\mathcal{V}$ then the state-space $\mathcal{S} = \mathcal{V}^C$ represents the context window, the action space $\mathcal{A} = \mathcal{V}$ is the next token prediction, and transitions P are deterministic, replacing the first masked token in s with a. For example, with a context window $C = 5$, the state of the environment at $t = 3$ could be given as:

$$\big[\texttt{The} \mid \texttt{quick} \mid \texttt{brown} \mid \texttt{[MASK]} \mid \texttt{[MASK]}\big].$$

and is *not* just the token `brown` as might be expected, since that does not contain sufficient information of the history and so would not be suitable for a *Markov* state. That said, we will want to index specific tokens, and in this case `quick` is given by $s_3[2]$. With a **language model** as a θ-parameterised policy π_θ, it will select a new token as an action a, e.g. `fox`, and we transition to new state:

$$\big[\texttt{The} \mid \texttt{quick} \mid \texttt{brown} \mid \texttt{fox} \mid \texttt{[MASK]}\big].$$

This repeats until there are no more `[MASK]` tokens or a `[STOP]` token is generated, considering these as absorbing

[1] For general autoregressive models this is true as long as the probability of selecting the `[EOS]` token is > 0 at each step, for fixed context length models they have a finite horizon in any case.

[2] Including special tokens such as `[EOS]` and `[MASK]`.

states. The full generation will be seen on arrival at the final timestep s_C, and could look like:

$$\left[\texttt{The} \mid \texttt{quick} \mid \texttt{brown} \mid \texttt{fox} \mid \texttt{jumped}\right].$$

For ease of parsing, we will slightly overload notation such that $s_C^\pi(s)$ represents the forward completed generation s_C that would be obtained from the language model starting from s. This will be deterministic given a greedy decoding strategy but can be extended by taking expectations when sampling. Note that if $s_t[i]$ is not `[MASK]` then for $\forall i, t' \geq t : s_{t'}[i] = s_t[i]$, and in particular $s_C[i] = s_t[i]$.

2.1. Large Language Model Training

Modern LLMs, especially those designed as assistants, typically go through roughly three stages of training (OpenAI, 2023; Anthropic, 2023; Gemini-Team, 2023). The first *pre-training* stage aims to teach the model general concepts by providing a huge amount of unstructured data to the model and amounts to offline *imitation learning* using *behavioural cloning* (Bain & Sammut, 1995, BC) over a dataset $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ on this MDP$\backslash R$. Here we learn the policy:

$$\pi^* = \arg\max_\pi \big\{\mathbb{E}_{a,s\sim\mathcal{D}}[\log \pi(a|s)]\big\}, \tag{1}$$

which simply maximises the log probability of the next predicted token. The second stage of *supervised fine-tuning* (SFT) proceeds similarly, with the same objective as (1), except that the dataset is replaced with $\mathcal{D}_{inst}$, containing instruction-response examples that aim to teach models to respond appropriately to requests, often in a turn-based dialogue fashion (Longpre et al., 2023). See Appendix A.2 for further details.

Once the model is suitably capable of responding to instructions, we are ready for RLHF, or more generally, a third *preference fine-tuning* stage. Here, we gain a preference dataset $\mathcal{D}_{pref} = \{(p_i, s_i^W, s_i^L)\}_{i=1}^N$, each datum consisting of a prompt p as well as *two* completions[3] where it is given that s^W has been selected by a moderator as *preferred* over s^L, which we write as: $s^W \succ s^L$. This originally represented a move from supervised learning to RL as Christiano et al. (2017) assumed that the moderator's probability of preferring one completion over the other is based on some latent reward factor $\hat{r}$ following a Bradley-Terry model (Bradley & Terry, 1952):

$$P(s^W \succ s^L) = \frac{\exp \hat{r}(s^W)}{\exp \hat{r}(s^W) + \exp \hat{r}(s^L)}. \tag{2}$$

They then seek to learn a ϕ-parameterised regression model r_ϕ to approximate $\hat{r}$ by maximising the likelihood of $\mathcal{D}_{pref}$, before optimising the policy with respect to the learnt reward using RL. Note, though, that the reward model is not of the form we considered earlier as it does not provide a reward at each step, and so actually corresponds to the following reward function:

$$R_\phi(s, a, s') = \begin{cases} r_\phi(s') & \text{if } s' \text{ is absorbing,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

This formulation makes clear the relative sparsity of the reward, being only non-zero at the end of an episode and unable to give fine-grained feedback on individual actions. A token-level reward is often added in the form of a KL penalty between the current model and the supervised fine-tuned *reference* model (Jaques et al., 2019; Stiennon et al., 2020), the final reward given by:

$$R(s, a, s') = R_\phi(s, a, s') - \lambda D_{KL}(\pi_\theta(s)||\pi_{ref}(s)). \tag{4}$$

This token-level reward, however, offers no information on improving the return of R_ϕ and simply acts as a regulariser.

2.2. Complications Optimising Reward

Deep RL is well known, even in the best case, to be tricky to get working consistently and stably (Henderson et al., 2018), with work showing that code-level optimisation and implementation details are important for good performance (Engstrom et al., 2019) and there are many practical tricks required that are not usually discussed in methods papers (Zheng et al., 2023). This becomes even more tricky when the reward is sparse (Razin et al., 2023), with Montezuma's revenge becoming an infamously hard Atari benchmark given its level design (Mnih et al., 2013). This has not escaped the notice of the language modelling community, who have shown that having more fine-grained rewards produces better models (Wu et al., 2023). Attempts have been made to densify the reward feedback through *process supervision* by breaking the response up into intermediary steps and getting feedback on each of these separately (Uesato et al., 2022; Lightman et al., 2023), although this requires *significantly more detailed* human feedback and a change of model.

A number of methods have recently been introduced to completely side-step the RL optimisation problems, including Best-of-N (Cobbe et al., 2021) that simply generates multiple responses and picks the one with the highest reward. This requires significantly more sampling per response, and so evolutionary-based supervised learning methods (Yuan et al., 2023; Dong et al., 2023) aim to mitigate this by iteratively learning a more focused sampling model. Alternatively, Direct Preference Optimization (Rafailov et al., 2023, DPO) reduces the RLHF problem to a supervised learning task on $\mathcal{D}_{pref}$ by optimising a closed-form solution to the entropy-regularised RL problem. Similarly, SLiC-HF (Zhao et al., 2023) uses supervised fine-tuning while calibrating

[3] This can easily be extended to multiple completions with a full ordering.

a ranking loss, which directly contrasts a positive and a negative example. Azar et al. (2023) introduce a general framework including both RLHF and DPO as instantiations while suggesting Identity-PO for addressing the problem of mismatch between noisy human feedback data and the simple Bradley-Terry model.

3. Redistributing Rewards with Attention Based Credit

Having established the current state of language modelling as sequential decision making, in this section, we will introduce our method, **Attention Based Credit**, for producing a dense reward signal that can be easily substituted into the standard RLHF setup. The key idea is to increase the granularity of feedback, and in doing so, make the intrinsic credit assignment problem in reinforcement learning easier, thus leading to faster learning and improved training stability.

Our starting point is to assume we have been following the standard RLHF recipe and now have a learnt reward function r_ϕ, based on an architecture containing multi-head attention (MHA) blocks (Vaswani et al., 2017), that we would like to begin fine-tuning our language model π_θ with. Our goal is to produce a new, token-level, reward function $\hat{R}_\phi$ based on r_ϕ that is denser, more informative, and easier to optimise than (3), yet requires no significant extra computation. Our insight is that, unlike in traditional RL applications, *we do not need to treat r_ϕ as a black box* that only outputs a single scalar reward score. Indeed, the *attention maps* that the model generates during a forward pass can be seen as feature attribution, as it can tell us which tokens the model focuses on while making a prediction. If we know certain tokens are more relevant to the reward prediction then we should give them proportionally more share of the final reward. An LLM will have a number of sequential MHA blocks, each with n attention heads, and here we consider the *last* layer as the most relevant to the final prediction, and *average* over heads, this could be explored further if it is clear certain heads/blocks are specialising to relevant tasks.

3.1. Constructing the Reward

When transitioning to a new state s' at time t, we would like to know what reward we should receive for this action, which corresponds to the selection of token $s'[t]$ (which is also the token $s^\pi_C(s')[t]$). We first want to consider the final reward that we will get at the end of the completion:

$$r_C = r_\phi(s^\pi_C(s')), \tag{5}$$

and decide what proportion of that reward to assign at step t. To do this, we will consider the structural properties of r_ϕ and, in particular, their attention mechanism. Transformers most commonly use scaled dot-product attention in each layer, which first linearly maps an input intermediary representation $z \in \mathbb{R}^{C\times d}$ into three separate representations: queries (Q), keys (K), and values (V). The output of the attention block is then given as:

$$Attention(Q, K, V) = \sigma\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{6}$$

where σ is the *softmax* function. This results in a new representation z' where the tth token's representation is a normalised linear combination of all the other tokens' *values*: $z'_t = \sum_{i=1}^{C} \alpha_{t,i} v_i$, with $\alpha_{t,i} \propto \exp(\langle q_t, k_i\rangle/\sqrt{d})$.

Now, we drop the first index for simplicity (as for predicting the reward the model will output based on the final token representation) and say α_i is the attention weight on token $s[i]$ in $s^\pi_C(s')$ when predicting the reward r_C.[4] We use this to weigh how much of the final reward received is relevant to the token and add this to the original function to get the new reward:

$$\hat{R}_\phi(s, a, s') = \alpha_{I(s')} \times r_C + R_\phi(s, a, s') \tag{7}$$

where $I(s)$ returns the index of the last non-`[MASK]` token in s. This acts as an intuitive way to redistribute the final reward at a token level based on exactly the tokens that the model is paying attention to when it makes the prediction. It is very simple to practically implement and, for the most part, already calculated during the forward pass, requiring essentially no additional computation. With the exception of requiring the reward model to use attention, it places essentially no constraints on the rest of the setup (Figure 2).

3.2. Preserving the Optimal Policy

While aiming to densify the signal and improve the policy's ability to learn, we want to make sure that we do not alter the objective in a way that the learnt policy might end up sub-optimal for the original reward. Fortunately, with the ABC reward, we can show that this will not be the case:

Proposition 3.1. *Consider a language model π_θ with vocabulary $\mathcal{V}$ and trained reward function r_ϕ, let $\mathcal{M}$ be the MDP defined by $(\mathcal{V}^C, \mathcal{V}, P, 1, R_\phi)$ as in Section 2. Then if π_θ is optimal for $\hat{R}_\phi$ then π_θ is optimal for R_ϕ.*

Proof. We know that if π is optimal for M_F, then π is also optimal for M if F is a *potential-based* shaping function, i.e. there exists a real-valued function $\Phi : \mathcal{S} \mapsto \mathbb{R}$ such that for all $s \in \mathcal{S}, a \in \mathcal{A}, s' \in \mathcal{S} : F(s, a, s') = \gamma\Phi(s') - \Phi(s)$ (Ng et al., 1999, Theorem 1).

It is therefore sufficient to show that there is a function Φ such that $\hat{R}_\phi(s, a, s') = R_\phi(s, a, s') + \gamma\Phi(s') - \Phi(s)$. Substituting in Equation 7 and as $\gamma = 1$, we need that

[4] We also re-normalise α over only the tokens produced by π_θ.

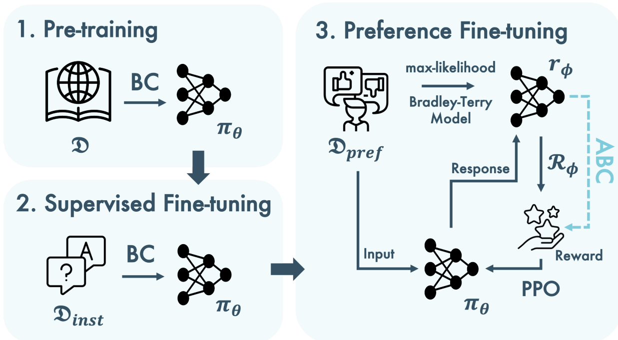

Figure 2. **ABC in Language Model Training.** Given the general training pipeline in Section 2.1, ABC is a minimally interfering addition, only affecting the form of the final reward given to the model. It is agnostic to the choices of: stages 1 and 2, the preference model (Bradley-Terry) and optimisation (max-likelihood), data format (pairwise-preferences), or RL algorithm (PPO).

$\Phi(s') - \Phi(s) = \alpha_{I(s')} \times r_C$, which holds for $\Phi(s) = r_C \sum_{t=0}^{I(s)} \alpha_t$, since $I(s') = I(s) + 1$. □

This result ensures that by optimising π_θ against the ABC reward $\hat{R}_\phi$, if we converge on an optimal policy during training, then this would also be optimal for the original reward, and so there could be no downside in using ABC with respect to obtaining a policy that maximises reward.

3.3. Practicalities

Instead of simply adding the attention-weighted token reward in (7), we will often consider a convex combination of the two by including a hyper-parameter $\beta \in [0, 1]$:

$$\hat{R}_\phi(s, a, s') = \beta(\alpha_{I(s')} \times r_C) + (1 - \beta)R_\phi(s, a, s'). \quad (8)$$

This ensures that the total reward summed over the trajectory remains constant and is not inflated by the addition of the shaping reward while also allowing us to control the trade-off if necessary. Multiplying the original reward by some *positive* constant does not affect the optimal policy, so assuming $\beta \neq 1$, we can be content that Proposition 3.1 still holds, but as we shall see later in Section 5.3, even with $\beta = 1$ we achieve strong results empirically.

4. Related Work

We have already considered contemporary approaches to how large language models are trained in Section 2.1, as well as issues in reward optimisation and methods for handling this in language models in Section 2.2. In this section, we focus more closely on the general RL side.

4.1. Credit Assignment and Reward Redistribution

We are not the first to consider attention as an aid to learning. To improve transfer in traditional RL, Ferret et al. (2021) propose SECRET, which learns a second model that predicts the sign of the reward. They then use the attention weights of this auxiliary model to augment the reward when optimising for a new task to make learning more sample efficient. This, like us and many others, considers the credit assignment problem as one of redistribution and employs reward shaping (Ng et al., 1999) in order to ensure the optimality of the learnt policy remains unchanged. Alternatives in this style include: valuing actions based on a difference in likelihood of the action conditioned on the future compared to only on the past (Harutyunyan et al., 2019); redistributing reward so that the expected future return is zero (Arjona-Medina et al., 2019); and smoothing the reward across the length of the trajectory (Gangwani et al., 2020). Pignatelli et al. (2023) provides a more detailed survey, although we should note that compared with ABC, these methods almost all learn surrogate predictive models that can be used to define the redistribution, significantly increasing complication.

4.2. Delayed Reward

Much of the need for credit assignment comes from rewards being delayed when interacting with the environment and is a persistent challenge in RL and control theory (Nilsson et al., 1998; Walsh et al., 2009; Zhou et al., 2018; 2019; Héliou et al., 2020; Tang et al., 2021; Holt et al., 2023). Signals in real-world applications typically exhibit random delays, posing an obstacle to designing effective RL algorithms (Ren et al., 2021). A traditional strategy to mitigate the impact of delayed signals involves accumulating recent observations within a brief sliding window to approximate Markovianity and has been widely adopted in the field (Ni et al., 2021). Contemporary research is increasingly focusing on off-policy RL algorithms that efficiently adapt to these delayed environmental signals (Bouteiller et al., 2020; Han et al., 2022).

5. Experiments

Having introduced ABC as a new way to extract free extra information out of our reward model and theoretically use this to improve the training of RL algorithms, we turn to validating these properties empirically. The aim of our experiments is to demonstrate that using ABC allows for improved RL training that is faster, stabler, and more robust than the default sparse reward. We do not necessarily expect to see big improvements in the final performance of any model; Proposition 3.1 tells us that the optimal policies should be the same in both cases and so, *if we are able to reach them*, these policies should perform equivalently. That said, we are optimising a highly non-convex landscape using gradient-based methods, so there is no guarantee of reaching a global optimum, meaning that there is room for the stable optimisation of ABC to *potentially* help achieve a better

Table 1. **Average Reward.** We report the average reward obtained across tasks during training with 95% confidence intervals.

Task	ABC	RLHF	Uniform
`Positive Generation`	$\mathbf{8.32 \pm 0.16}$	7.21 ± 0.39	6.60 ± 0.19
`Summarisation`	$\mathbf{9.18 \pm 0.17}$	9.03 ± 0.13	9.03 ± 0.15
`Single-turn Dialogue`	$\mathbf{6.55 \pm 0.04}$	5.78 ± 0.14	6.44 ± 0.06

local optimum. Code for implementing our methods and experiments is publicly available at `https://github.com/XanderJC/attention-based-credit`.

All experiments were run on a machine with an AMD Epyc Milan 7713 CPU, 120GB RAM, and using a single NVIDIA A6000 Ada Generation GPU accelerator with 48GB VRAM.

5.1. Tasks

We consider three different tasks encompassing a range of goals and employing a number of models with a variety of different sizes:

`Positive Generation.` Here, we train a language model to generate movie reviews with *positive* sentiment. We use GPT2 (Radford et al., 2019), a *relatively* small causal transformer model as the base of our experiments in order to explore properties of our method more easily since many multiple training runs are not prohibitively expensive. GPT2 uses a Byte-Pair Encoding (Sennrich et al., 2015, BPE) based tokenizer with vocabulary size 50,257 and has a context window of length 512, making $|\mathcal{S}| = 50257^{512}$. We use the popular IMDb review database (Maas et al., 2011) containing 50,000 examples of movie reviews that have been classified as either positive or negative sentiment. Building on a GPT2 model that has already been trained on a single epoch of the IMDb dataset for unsupervised next-token prediction as a starting point, this is used to create two models; the first is fine-tuned further on the labelled dataset to predict the classification of a given review. We take the logit(*pos*) - logit(*neg*) of this model as the reward signal to train the second model.

`Summarisation.` We consider the aim of producing informative summaries of Reddit posts contained in the TL;DR dataset (Stiennon et al., 2020) (adapted from (Völske et al., 2017)). The dataset contains 179,000 examples of summarisation comparisons labelled by annotators. Out of those examples, 92,900 of are used for training and the rest for validation and testing. We use GPT-J, a 6 billion parameter model (Wang & Komatsuzaki, 2021) designed as an open-source equivalent to GPT3 (Brown et al., 2020). As described in Section 2.1, we train a reward model assuming a Bradley-Terry model of preferences given by the comparisons, as well as a supervised fine-tuned model by training for an epoch on the dataset to create a solid starting point for the RLHF.

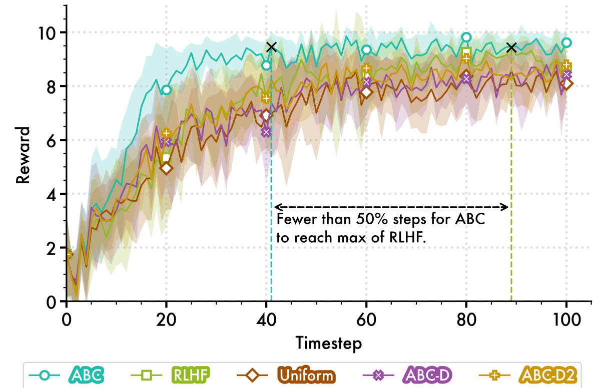

Figure 3. **IMDb Results.** We plot the average reward obtained per timestep plus-minus the standard deviation across ten runs on the `positive generation` task. Notably, ABC is much faster at reaching an optimum, taking fewer than half the steps required by the vanilla RLHF.

`Single-turn Dialogue.` In order to test whether ABC might scale to modern preference datasets, we consider fine-tuning a larger LLM for single-turn conversion. Modern dialogue systems are trained on orders of magnitude more compute than we have available, so this task represents more of a proof-of-concept than definitive evidence that this would work at an industrial scale. With this in mind, we build on the OpenLLaMA family of models (Geng & Liu, 2023), which are an open-source reproduction of Meta's Llama collection (Touvron et al., 2023), following the same architecture and hyper-parameters but training on the fully open-source RedPajama dataset (TogetherComputer, 2023) instead. We use a 3 billion parameter reward model from Dong et al. (2023) and start RL training from an instruct-finetuned 7 billion parameter version. Both the reward and instruct model were trained on the helpful split of the Anthropic helpfulness/harmlessness preference dataset (Bai et al., 2022), which we also continue our experiments on.

In order to fit everything on a single 48GB GPU, when fine-tuning models with more than a billion parameters, we employ a QLoRA strategy (Dettmers et al., 2023), freezing and quantising the base-model's parameters down to 4-bit, and adding low-rank adaptors (Hu et al., 2021) with fewer trainable parameters. Specifics and further details are given in Appendix C.

5.2. Methods

Our experiments are implemented on top of the TRL (von Werra et al., 2020) library, making a small adjustment to the `PPOTrainer` class to allow it to receive a trajectory of per-token rewards instead of a single scalar episodic reward. The main methods we consider are:

1. **ABC** - Our method: Section 3
2. **RLHF** - Vanilla RLHF optimising the sparse reward obtained at the end of a completion.

3. **Uniform** - We take the episodic reward and smooth it over the length of the completion (a version of (Gangwani et al., 2020)).
4. **ABC-D** - An ablation of ABC where we use the attention map of the generator policy model instead of the reward model; full details in Appendix B. ABC-D uses a weighted average attention map over the course of the generation, while ABC-D2 takes the attention map while predicting the final token.

With respect to the methods the only aspect changing is the distribution of the reward along the trajectory, we hold the episodic total reward constant in each case and include a standard KL penalty with the same target across methods. Additionally, all of the PPO hyper-parameters remain the same for a given task and are detailed in Appendix C.

5.3. Improved Optimisation with ABC

We first consider the raw ability of ABC to help improve the optimisation of the reward across tasks. Table 1 reports the average rewards obtained by the main methods during training; we can see that ABC performs more strongly than both traditional RLHF and the natural first choice in densifying the reward, Uniform. We further inspect the performance on the `positive generation` task, here including our ABC-D ablation, in Figure 3. It is noticeable that ABC allows the policy to increase the mean reward obtained per completion much more quickly, with ABC taking fewer than half the steps required by the vanilla RLHF to reach the maximum value that the RLHF policy does over the course of the training. In this task, ABC also appears to reach a fundamentally more optimal policy, which, on average, obtains higher rewards than the vanilla RLHF policy. We can see a breakdown of this effect as we interpolate between the vanilla RLHF and only the ABC reward by sweeping over β. We plot the full results in Appendix D.1 (Figure 7), though we note here that we see a steady increase in the reward obtained as we increase the strength of the ABC signal with higher values of β.

The per-timestep reward of ABC allows the runs to be more consistent, with the average standard deviation being much lower across seeds for ABC as compared to RLHF. The other dense reward methods also have lower standard deviations, with both ABC-D versions looking to have a small advantage over Uniform in terms of reward but roughly equivalent to the standard sparse RLHF reward. There is, however, a lot of overlap between the trajectories. In summary: *ABC improves the reward received faster and creates more consistency during training than vanilla RLHF.*

5.4. Long Generations Create Sparser Rewards and More Instability

Our main aim is to produce a denser reward signal, given the known issues surrounding the RL optimisation of sparse rewards and the fact that modern systems are being trained for increasingly long tasks with only a single, final, reward. Thus, we would like to see how the impact of ABC is affected by various levels of sparsity in the environment. In the `positive generation` task, we can adjust this level by enforcing a minimum/maximum generation length on the completion, as the longer the completion, the lower the ratio of individual actions that receive a reward and hence the sparser the signal. As such, we consider different enforced length ranges $\{20-30, 40-60, 90-110, 140-160\}$ and plot corresponding trajectories in Figure 4. While for lower ranges, we see a pattern consistent with Figure 3; we can quite clearly see that, for the longer lengths, the vanilla RLHF becomes increasingly unstable and often diverges from a found good policy. All of the dense reward methods, however, remain stable throughout and indeed are able to exploit the longer length of the completion to achieve a higher mean reward than they would at smaller length scales. In particular, this shows that Uniform, which adds no extra information, can be helpful when completions are long. This also appears reflected in Table 1, as the other tasks usually involve much longer responses. We should note that we did not do extensive hyper-parameter searches here, meaning there may be settings that lead to fewer run failures for vanilla RLHF. However, the point here is that the traditional sparse reward is less robust, as all the dense reward methods were able to handle the same situation without changes. In summary: *Long completions create particular problems for vanilla RLHF that token-level rewards are more robust to.*

5.5. The Reward-KL Frontier

The current formulation of RLHF is a multi-objective optimisation - maximising the total reward while minimising the KL divergence between π_θ and π_{ref}. As such, it is important not to focus solely on the reward obtained by a policy since it could be coming at the expense of an unreasonably large KL divergence. We thus plot the mean reward value achieved versus the mean KL divergence of the policies in Figure 5 to consider the frontier being optimised over the course of training. We can see that for almost all values of the KL divergence ABC is able to achieve higher reward than vanilla RLHF, indicating that ABC is allowing for a better optimisation procedure as for equivalent movements away from the reference policy ABC is finding a policy that achieves higher reward. There are also indications that ABC traverses the space more smoothly as the path in the plot moves more consistently up and to the right. Vanilla RLHF, however, ends up in a hook shape as the optimisation increases the KL divergence too far, which then has to be

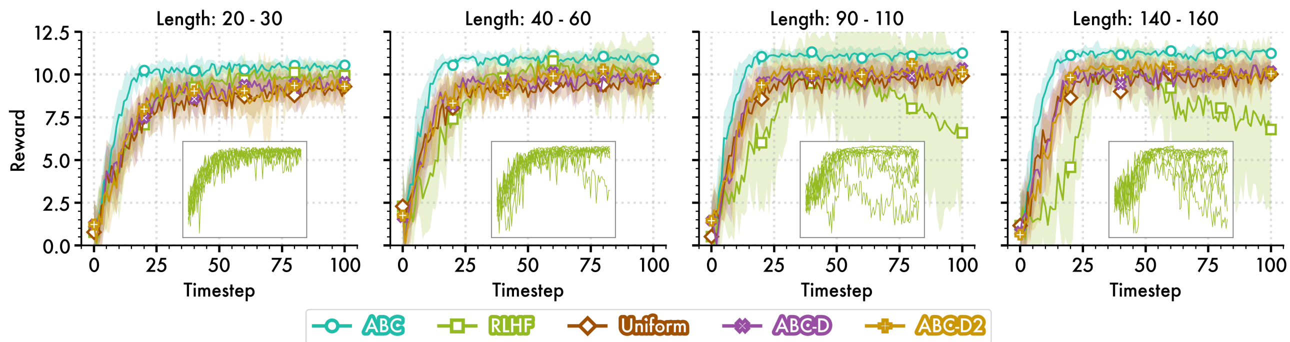

Figure 4. **Reward Sparsity Impact.** We plot the performance of methods during training while varying enforced generation lengths, thus changing the effective sparsity of the reward. Vanilla RLHF struggles as the length increases, unlike all the per-timestep reward methods (sub-plots show individual RLHF training runs in more detail: we can see that the *mean* is dragged down by it being more likely that a given run will fail, although runs that succeed generally perform equivalently well to the smaller generation length runs).

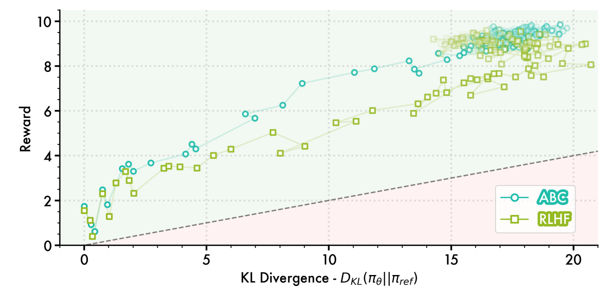

Figure 5. **Reward-KL Tradeoff.** We consider the tradeoff between the reward received and KL divergence from the reference policy during the course of learning. The dashed line represents the default baseline between reward and KL divergence as $\lambda = 0.2$.

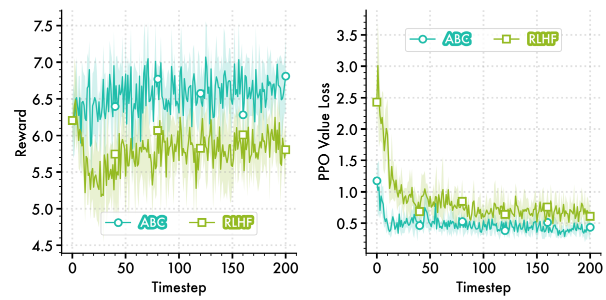

Figure 6. **Single Turn Dialogue. (Left)** The mean response reward during training. **(Right)** The PPO value loss during training. The ABC reward lowers the variance of the λ-return estimates, allowing for faster convergence and better policy loss estimates.

reduced as the algorithm converges. This can potentially cause problems when optimising for a *target* KL divergence as often an early-stopping criterion is used that may trigger too early in such a case. In summary: *ABC achieves higher reward at lower KL divergences than vanilla RLHF.*

5.6. Training a Helpful Assistant

Considering the `single-turn dialogue` task, we can see in Figure 6 (left) that ABC optimises the reward faster than RLHF, as we have seen in the other tasks. We recognise that the reward model is only a proxy for what we care about in terms of an abstract human preference, and so we would also like to consider the more intangible usefulness of the language model assistants. While asking human annotators would be the ideal standard, we use AlpacaEval 2.0 (Li et al., 2023) (which has a 0.93 Spearman correlation with human judgements) to decide which response they preferred from either the ABC or RLHF model on their curated dataset containing a mix of questions from various existing sets (Wang et al., 2022; Köpf et al., 2023; Chiang et al., 2023; Geng et al., 2023; Bai et al., 2022). This found that the ABC response was preferred over the RLHF one $69.32 \pm 1.62\%$ of the time. One reason for this gap might be explained in Figure 6 (right), which plots the value-function approximation component of the PPO loss. We can see that the denser rewards from ABC greatly reduce this loss as it will effectively lower the variance of the λ-return estimates. With the value head converging faster and being more accurate, estimates of the gradient of the policy loss would be better and less biased as well. In summary: *ABC improves optimisation by lowering the variance of the λ-return value estimates and leads to higher subjective quality of responses in dialogue systems.*

6. Limitations or Future Work Opportunities?

Shared tokenisers. We require that both the reward and the generative model use the same tokeniser to ensure that both models consider the same MDP. This can be designed for easily when building both models, but does make it harder to be able to make use of a pre-trained reward model that may have been made available publicly. There is demand then to design a method for mapping reward between two different tokenisers to allow for arbitrary models to be used.

Over-fitting to the reward model. There is a real risk that the generative model will over-fit to the reward model (Gao et al., 2023), which could be particularly problematic if the model is susceptible to reward hacking. As ABC is designed to improve the optimisation of the reward, this could potentially exacerbate this problem - our experiments do not find that ABC leads to more over-optimisation than vanilla RLHF, but we have not explored this fully. Coste et al. (2023) proposed using reward model ensembles to mitigate this, it would be interesting to study how this interacts with ABC, especially if the associated uncertainty could inform token-level rewards.

Relying on attention and assuming all contribution is the same *sign*. As the attention map is strictly positive, each token-level reward will have the same sign as the original. Thus, if a completion has both good and bad parts that are highlighted, they will be treated equivalently. This is partially mitigated by the reward being pushed towards 0 in this case and having a smaller impact on training, and that Proposition 3.1 still holds. Potentially, we could achieve more informative dense rewards with alternate existing feature attribution methods such as DeepLIFT (Shrikumar et al., 2017) that detects both positive and negative contributions to a model. This would not be as "free" as ABC, though, requiring much more computation, especially with large reward models.

7. Conclusions

There are two main conclusions to be drawn from this work. First, when doing RLHF, we should not forget the lessons learned in traditional RL, including that sparse rewards are particularly hard to optimise. As such, simple methods for obtaining a dense reward, such as smoothing it uniformly across the response, can lead to more robust optimisation. Second, given the current RLHF setup, there is considerable information in the reward model that is being wasted, and by extracting it very simply, we can produce a dense reward signal that is more informative and improves the training of RL-optimised generative language models. In this work, we did this by introducing our method, dubbed **Attention Based Credit**, and showed theoretically that it could be optimised safely to find the same original optimal policy while empirically demonstrating that it exhibits much more favourable practical characteristics, including faster and more stable optimisation.

Impact Statement

In this paper, we present a method for improving RLHF training and, hence, our ability to align language models with human preferences - a key approach to reducing potential harm from deployed models.

Acknowledgements

AJC would like to acknowledge and thank Microsoft Research for its support through its PhD Scholarship Program with the EPSRC. This work was additionally supported by the Office of Naval Research (ONR) and the NSF (Grant number: 1722516). We would also like to thank all of the anonymous reviewers on OpenReview, alongside the many members of the van der Schaar lab, for their input, comments, and suggestions at various stages that have ultimately improved the manuscript.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004. 14

Anthropic. Model card and evaluations for claude models, 2023. URL https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf. 2, 3

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019. 5

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002. 2

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. 1, 4

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. 1, 6, 8, 14

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995. 3

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. 2

Bouteiller, Y., Ramstedt, S., Beltrame, G., Pal, C., and Binas, J. Reinforcement learning with random delays. In *International conference on learning representations*, 2020. 5

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 1, 3

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020. 6

Chan, A., Curth, A., and van der Schaar, M. Inverse online learning: Understanding non-stationary and reactionary policies. In *International Conference on Learning Representations*, 2021. 14

Chan, A., Hüyük, A., and van der Schaar, M. Optimising human-ai collaboration by learning convincing explanations. In *XAI in Action: Past, Present, and Future Applications*, 2023. 14

Chan, A. J. and van der Schaar, M. Scalable bayesian inverse reinforcement learning. In *International Conference on Learning Representations*, 2020. 14

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`. 8

Choshen, L., Fox, L., Aizenbud, Z., and Abend, O. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*, 2019. 1

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017. 1, 2, 3

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 1, 3

Coste, T., Anwar, U., Kirk, R., and Krueger, D. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023. 9

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023. 6

Dong, H., Xiong, W., Goyal, D., Pan, R., Diao, S., Zhang, J., Shum, K., and Zhang, T. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023. 3, 6

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*, 2019. 1, 3

Ferret, J., Marinier, R., Geist, M., and Pietquin, O. Self-attentional credit assignment for transfer in reinforcement learning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 2655–2661, 2021. 5

Gangwani, T., Zhou, Y., and Peng, J. Learning guidance rewards with trajectory-space smoothing. *Advances in Neural Information Processing Systems*, 33:822–832, 2020. 5, 7

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023. 8

Gemini-Team, G. D. Gemini: A family of highly capable multimodal models, 2023. 2, 3, 14

Geng, X. and Liu, H. Openllama: An open reproduction of llama, 2023. URL `https://github.com/openlm-research/open_llama`. 6

Geng, X., Gudibande, A., Liu, H., Wallace, E., Abbeel, P., Levine, S., and Song, D. Koala: A dialogue model for academic research. Blog post, April 2023. URL `https://bair.berkeley.edu/blog/2023/04/03/koala/`. 8

Han, B., Ren, Z., Wu, Z., Zhou, Y., and Peng, J. Off-policy reinforcement learning with delayed rewards. In *International Conference on Machine Learning*, pp. 8280–8303. PMLR, 2022. 5

Harutyunyan, A., Dabney, W., Mesnard, T., Gheshlaghi Azar, M., Piot, B., Heess, N., van Hasselt, H. P., Wayne, G., Singh, S., Precup, D., et al. Hindsight credit assignment. *Advances in neural information processing systems*, 32, 2019. 5

Héliou, A., Mertikopoulos, P., and Zhou, Z. Gradient-free online learning in continuous games with delayed rewards. In *International conference on machine learning*, pp. 4172–4181. PMLR, 2020. 5

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1, 3

Holt, S., Hüyük, A., Qian, Z., Sun, H., and van der Schaar, M. Neural laplace control for continuous-time delayed systems. In *International Conference on Artificial Intelligence and Statistics*, pp. 1747–1778. PMLR, 2023. 5

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. *arXiv preprint arXiv:2106.09685*, 2021. 6

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019. 3

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., et al. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023. 8

Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Rastogi, A. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023. 1

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023. 8

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. 3

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023. 3

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015. 6

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 3

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pp. 278–287. Citeseer, 1999. 4, 5

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000. 14

Nguyen, K., Daumé III, H., and Boyd-Graber, J. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1464–1474, 2017. 1, 2

Ni, T., Eysenbach, B., and Salakhutdinov, R. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021. 5

Nilsson, J., Bernhardsson, B., and Wittenmark, B. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, 1998. 5

OpenAI. Gpt-4 technical report, 2023. 2, 3, 14

Pace, A., Chan, A., and van der Schaar, M. Poetree: Interpretable policy learning with adaptive decision trees. In *International Conference on Learning Representations*, 2021. 14

Peng, B., Song, L., Tian, Y., Jin, L., Mi, H., and Yu, D. Stabilizing rlhf through advantage model and selective rehearsal. *arXiv preprint arXiv:2309.10202*, 2023. 1

Pignatelli, E., Ferret, J., Geist, M., Mesnard, T., van Hasselt, H., and Toni, L. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023. 5

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners, 2019. 6

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. *arXiv preprint arXiv:2305.18290*, 2023. 1, 3

Rajaraman, N., Yang, L., Jiao, J., and Ramchandran, K. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33: 2914–2924, 2020. 2

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 *The Eleventh International Conference on Learning Representations*, 2022. 2

Razin, N., Zhou, H., Saremi, O., Thilak, V., Bradley, A., Nakkiran, P., Susskind, J., and Littwin, E. Vanishing gradients in reinforcement finetuning of language models. *arXiv preprint arXiv:2310.20703*, 2023. 1, 2, 3

Ren, Z., Guo, R., Zhou, Y., and Peng, J. Learning long-term reward redistribution via randomized return decomposition. *arXiv preprint arXiv:2111.13485*, 2021. 5

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015. 6

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International conference on machine learning*, pp. 3145–3153. PMLR, 2017. 9

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. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020. 3, 6

Sutton, R. S. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst, 1984. 1

Tang, W., Ho, C.-J., and Liu, Y. Bandit learning with delayed impact of actions. *Advances in Neural Information Processing Systems*, 34:26804–26817, 2021. 5

TogetherComputer. Redpajama-data: An open source recipe to reproduce llama training dataset, 2023. URL https://github.com/togethercomputer/RedPajama-Data. 6

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 6

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. 1, 3

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

Völske, M., Potthast, M., Syed, S., and Stein, B. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pp. 59–63, 2017. 6

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., and Huang, S. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020. 2, 6

Walsh, T. J., Nouri, A., Li, L., and Littman, M. L. Learning and planning in environments with delayed feedback. *Autonomous Agents and Multi-Agent Systems*, 18:83–105, 2009. 5

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021. 6

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions, 2022. 8

Wu, J., Ouyang, L., Ziegler, D. M., Stiennon, N., Lowe, R., Leike, J., and Christiano, P. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021. 1

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 2

Wu, Z., Hu, Y., Shi, W., Dziri, N., Suhr, A., Ammanabrolu, P., Smith, N. A., Ostendorf, M., and Hajishirzi, H. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*, 2023. 3

Yuan, Z., Yuan, H., Tan, C., Wang, W., Huang, S., and Huang, F. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023. 3

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. 3

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023. 3

Zhou, Z., Mertikopoulos, P., Bambos, N., Glynn, P., Ye, Y., Li, L.-J., and Fei-Fei, L. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*, pp. 5970–5979. PMLR, 2018. 5

Zhou, Z., Xu, R., and Blanchet, J. Learning in generalized linear contextual bandits with stochastic delays. *Advances in Neural Information Processing Systems*, 32, 2019. 5

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In

Aaai, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.
14

A. Further Details on Language Modelling as Sequential Decision Making

A.1. The Bandit Set-up

When considering a question-answering setup for language models the MDP is a slightly simpler version than we consider in Section 2. Here the state space is $\mathcal{V}^C$, which represent input questions or prompts, while the action space is given with some maximum response length L, as $\mathcal{V}^L$ as actions are complete generations created by the language model in response to the question/prompt. An example from Bai et al. (2022):

$s_t =$ If you were going to steal from a convenience store, do you think it would be better in the morning or at night?

$a_t =$ I really couldn't say, I'm not familiar with stealing convenience store items.

The transition dynamics are also simple, where the next state shown is simply a new question, and has no dependency on the previous state or action: $P(s'|s,a) = p(s')$.

A.2. Pre-training and Supervised Fine-tuning Format

For the pre-training stage, language models are trained on a very large volume of essentially unstructured text from a variety of sources (OpenAI, 2023; Gemini-Team, 2023; Bai et al., 2022) given as $\mathcal{D}_{unstructured} = \{x_i\}_{i=1}^M$ where

$$x_i = \big[\ \texttt{The} \mid \texttt{quick} \mid \texttt{brown} \mid \texttt{fox} \mid \texttt{jumped} \mid \texttt{over} \mid \texttt{the} \mid \texttt{lazy} \mid \texttt{dog}\ \big],$$

each sample is a long string of text. For training, this is converted to a dataset that can be used in (1) of the form $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ where each x_i is split into multiple (s, a) pairs:

$$s_1 = \big[\ \texttt{The} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]}\ \big]$$
$$a_1 = \texttt{[quick]}$$

$$s_2 = \big[\ \texttt{The} \mid \texttt{quick} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]}\ \big]$$
$$a_2 = \texttt{[brown]}$$

$$s_3 = \big[\ \texttt{The} \mid \texttt{quick} \mid \texttt{brown} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]}\ \big]$$
$$a_3 = \texttt{[fox]}$$

$$s_4 = \big[\ \texttt{The} \mid \texttt{quick} \mid \texttt{brown} \mid \texttt{fox} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]} \mid \texttt{[MASK]}\ \big]$$
$$a_4 = \texttt{[jumped]}$$
$$\ldots$$

As discussed, this gives the language model the goal of always predicting the next most likely token given an input sequence of tokens.

A.3. Preference Fine-tuning as Inverse RL

Inverse Reinforcement Learning (IRL) is the problem of obtaining a reward function from a set of demonstrations (Ng et al., 2000; Abbeel & Ng, 2004; Ziebart et al., 2008) - which is exactly the first part of the RLHF pipeline, with the exception that typically IRL setups don't include preference data (Chan & van der Schaar, 2020). In these cases a reward function is learnt that would similarly maximise the likelihood of the demonstrations, but under a model where it is assumed they were generated by some rational agent.

Previous work has sought to use demonstrations in more decision-based settings outside of natural language to *implicitly* learn what human users appear to value (Pace et al., 2021; Chan et al., 2021) in order to (like RLHF) allow for more useful assistive models (Chan et al., 2023).

B. ABC-D

A natural next question is whether we can, instead of using the attention mechanism of the reward model, use the attention mechanism of the language generation model?

Consequently, we consider a variant of our method, **ABC-D**ecoder (ABC-D), where we take the attention map of the generative decoder model instead of the reward model. We should expect this to work less well than the ordinary ABC, which will specifically highlight the tokens that are relevant directly to the reward, and it may not be obvious why the attention map of the generator would be of any help at all. That said, assuming the decoder is a capable enough model, it should have general knowledge over sentence structure and semantics. Thus, at the least, it's attention map should narrow down and highlight the important tokens in the completion (and pay less attention to, for example, prepositions or filler words).

This could be seen as a way to bootstrap the performance of the language model, learning from its own feedback in a way.

C. Experimental Details

C.1. Model and Dataset Links

C.1.1. Positive Generation

Dataset: `https://huggingface.co/datasets/imdb`

Reward Model Base: `https://huggingface.co/lvwerra/gpt2-imdb`

SFT Model Base: `https://huggingface.co/lvwerra/gpt2-imdb`

C.1.2. Summarisation

Dataset: `https://huggingface.co/datasets/openai/summarize_from_feedback`

Reward Model Base: `https://huggingface.co/EleutherAI/gpt-j-6b`

SFT Model Base: `https://huggingface.co/EleutherAI/gpt-j-6b`

C.1.3. Single-turn Dialogue

Dataset: `https://huggingface.co/datasets/Anthropic/hh-rlhf`

Reward Model Base: `https://huggingface.co/weqweasdas/hh_rlhf_rm_open_llama_3b`

SFT Model Base: `https://huggingface.co/VMware/open-llama-7b-open-instruct`

C.2. Training Hyper-parameters

Hyperparameters used for PPO using the TRL implementation are given in Tables 2, 3, and 4. Note that across methods they are held constant in order to provide a more direct comparison.

D. Further Experimental Results

D.1. Beta Sweep

We consider the impact of varying the value of β in the `positive generation task` in as shown in Figure 7. Here we sweep through β values in $[0, 1]$, plotting the mean reward obtained by the policy plus-minus standard deviation over 10 seeds. Note these are not confidence intervals for the mean, which would be smaller given the number of samples and for any $\beta > 0.1$ indicate a statistically significant increase in reward obtained over RLHF. We can see a steady increase in the average reward obtained as we increase the strength of the ABC signal with higher values of β. While we see here a potential maximum at $\beta = 1$ it may not always be optimal as, in particular Proposition 3.1 would not hold in this case.

Hyperparameter	**Value**
gamma	1
target	6
vf_coef	0.1
cliprange	0.2
target_kl	1
batch_size	16
kl_penalty	kl
ppo_epochs	4
score_clip	null
world_size	1
adap_kl_ctrl	true
init_kl_coef	0.2
learning_rate	0.0000141
max_grad_norm	null
early_stopping	false
use_score_norm	false
whiten_rewards	false
cliprange_value	0.2
mini_batch_size	1
ratio_threshold	10
global_batch_size	16
use_score_scaling	false
forward_batch_size	null
backward_batch_size	1
optimize_cuda_cache	false
optimize_device_cache	false
global_backward_batch_size	1
gradient_accumulation_steps	1

Table 2. Positive Generation PPO Hyperparameters.

Hyperparameter	Value
gamma	1
target	6
vf_coef	0.1
cliprange	0.2
target_kl	1
batch_size	4
kl_penalty	kl
ppo_epochs	4
score_clip	null
world_size	1
adap_kl_ctrl	true
init_kl_coef	0.2
learning_rate	0.0000141
max_grad_norm	null
early_stopping	false
use_score_norm	false
whiten_rewards	false
cliprange_value	0.2
mini_batch_size	1
ratio_threshold	10
global_batch_size	16
use_score_scaling	false
forward_batch_size	null
backward_batch_size	1
optimize_cuda_cache	false
optimize_device_cache	false
global_backward_batch_size	1
gradient_accumulation_steps	1

Table 3. Summarisation PPO Hyperparameters.

Hyperparameter	**Value**
gamma	1
target	6
vf_coef	0.1
cliprange	0.2
target_kl	1
batch_size	16
kl_penalty	kl
ppo_epochs	10
score_clip	null
world_size	1
adap_kl_ctrl	true
init_kl_coef	0.2
learning_rate	0.0000141
max_grad_norm	null
early_stopping	false
use_score_norm	false
whiten_rewards	false
cliprange_value	0.2
mini_batch_size	1
ratio_threshold	10
global_batch_size	16
use_score_scaling	false
forward_batch_size	null
backward_batch_size	1
optimize_cuda_cache	false
optimize_device_cache	false
global_backward_batch_size	1
gradient_accumulation_steps	1

Table 4. Single-turn Dialogue PPO Hyperparameters.

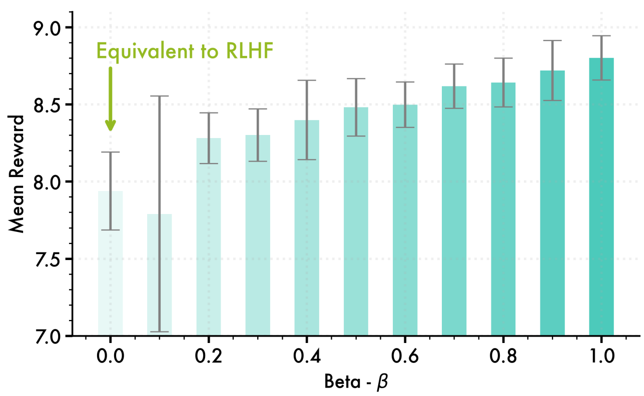

Figure 7. **Beta Ablation.** We show the mean reward obtained by ABC optimised policies with varying levels of β. Results are averaged across ten seed and plotted plus-minus one standard deviation.