BRAIn: Bayesian Reward-conditioned Amortized INference for natural language generation from feedback

Gaurav Pandey [1] Yatin Nandwani [1] Tahira Naseem [1] Mayank Mishra [1] Guangxuan Xu [1] Dinesh Raghu [1] Sachindra Joshi [1] Asim Munawar [1] Ramón Fernandez Astudillo [1]

Abstract

Distribution matching methods for language model alignment such as Generation with Distributional Control (GDC) and Distributional Policy Gradient (DPG) have not received the same level of attention in reinforcement learning from human feedback (RLHF) as contrastive methods such as Sequence Likelihood Calibration (SLiC), Direct Preference Optimization (DPO) and its variants. We identify high variance of the gradient estimate as the primary reason for the lack of success of these methods and propose a self-normalized baseline to reduce the variance. We further generalize the target distribution in DPG, GDC and DPO by using Bayes' rule to define the reward-conditioned posterior. The resulting approach, referred to as BRAIn - **B**ayesian **R**eward-conditioned **A**mortized **In**ference acts as a bridge between distribution matching methods and DPO and significantly outperforms prior art in summarization and Antropic HH tasks.

1. Introduction

Reinforcement Learning from Human Feedback (RLHF) has emerged as a pivotal technique to fine-tune Large Language Models (LLMs) into conversational agents that obey pre-defined human preferences (Ouyang et al., 2022; Bai et al., 2022; OpenAI et al., 2023; Touvron et al., 2023). This process involves collecting a dataset of human preferences and using it to align a Supervised Fine-Tuned (SFT) model to human preferences.

Proximal Policy Optimization (PPO) RLHF (Ziegler et al., 2019), has been instrumental in the development of groundbreaking models such as GPT-3.5 (Ouyang et al., 2022; Ye et al., 2023) and GPT-4 (OpenAI et al., 2023) among others. This RL technique trains a separate Reward Model (RM) to discriminate outputs based on human preferences. The RM is then used to train a policy that maximizes the expected reward while regularizing it to not diverge too much from the SFT model.

Despite its clear success, PPO has lately been displaced by offline contrastive techniques that are more scalable and do not make full use of a separate RM. Techniques like Likelihood Calibration (SLiC) (Zhao et al., 2023) or Rank Responses to align Human Feedback (RRHF) (Yuan et al., 2023) only keep ranking information from a RM. Direct Preference Optimization (DPO) (Rafailov et al., 2023), which is currently the de-facto method used to align high-performing models such as Zephyr (Tunstall et al., 2023), Mixtral (Jiang et al., 2024) or LLaMa-3[1], trains the policy directly on human preferences without the need of a separate reward model[2].

Both PPO and DPO are derived from KL-controlled reward maximization (Jaques et al., 2017), which has a well known closed form solution (Levine, 2018). This optimal policy comes however in the form of an energy-based model (EBM) with an intractable partition function. A set of less well-known methods for RL in language models use distribution matching to align an SFT model to this EBM (Khalifa et al., 2020b; Parshakova et al., 2019; Korbak et al., 2022). During the alignment of an SFT model via distribution matching, we need to sample from the target EBM. However, sampling from the target EBM is challenging, and hence distribution matching approaches sample from a proposal distribution instead, and reweigh the samples based on their importance weights. Despite the clear intuition behind distribution matching, these methods are not used commonly for the task of reinforcement learning from human feedback.

In this work, we address the primary reason for the lack of success of distribution matching methods for RLHF. Towards this end, we propose BRAIn- **B**ayesian

[1]IBM Research AI. Correspondence to: Gaurav Pandey <gpandey1@in.ibm.com>.

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

[1]https://ai.meta.com/blog/meta-llama-3/

[2]Arguably, the more performant versions of DPO still depend on a RM for determining preference data (Liu et al., 2023).

Reward-conditioned **A**mortized **In**ference that extends the distribution-matching methods in two significant ways. Firstly, we propose a Bayesian approach to construct the target distribution for distribution matching. Specifically, we treat the SFT model as the prior distribution over the outputs for a given input, that is, $p(y|x)$. The likelihood $p(G = 1|x, y)$ captures the goodness of an output for a given input and is defined as a function of the reward $r(x, y)$ based on the reward-modelling assumptions. The resulting reward-conditioned posterior $p(y|x, G = 1)$, obtained using Bayes' rule, is chosen as the target distribution. When the underlying preference model behind the reward function $r(x, y)$ follows Bradley-Terry assumptions, we show that the posterior corresponds to the optimal policy of the KL-regularized reward-maximization objective used in PPO-RLHF (Ziegler et al., 2019).

More significantly, we observe that the distribution matching technique suffers from high-variance of the gradient estimate, despite the baseline proposed in Korbak et al. (2022). In this work, we propose a self-normalized baseline that significantly reduces the variance of the gradient estimate. We prove that the resulting estimate corresponds to the gradient of a novel self-normalized KL divergence objective for distribution matching. Furthermore, self-normalization in the baseline helps us to establish DPO-sft (a version of DPO where samples are generated from the SFT model and scored by the reward model) as a special case of the BRAIN objective.

In our experiments on TL;DR summarization (Stiennon et al., 2020; Völske et al., 2017) and AntropicHH (Bai et al., 2022), BRAIN establishes a new state-of-the-art by significantly outperforming existing SoTA RL methods DPO and RSO (Rafailov et al., 2023). In addition, we bridge the gap between BRAIN and DPO by careful augmentation of DPO objective in two ways. Specifically, we incorporate: 1) multiple outputs for a given input prompt, and 2) reward as an importance weight in the DPO objective.

Overall, we make the following contributions:

1. We propose a Bayesian approach to construct the target distribution in distribution matching methods of RL for LLMs. The resulting posterior generalizes the PPO optimal policy.
2. For distilling the posterior in our training policy, we propose a self-normalized baseline for variance reduction of the gradient estimate of the objective. The resulting algorithm is referred to as BRAIN and the gradient estimate is referred to as the BRAIN gradient estimate.
3. We theoretically prove that the proposed gradient estimate is an unbiased estimator of a modified form of KL divergence that we name BRAIN objective.
4. We derive the exact form of the BRAIN objective under Bradley-Terry preference model assumption. We also show DPO can be derived as a special case of the BRAIN objective.
5. Finally, we empirically substantiate our claims by experimenting on two natural language generation tasks.

2. Related Works

In relation to RLHF approaches, InstructGPT (Ouyang et al., 2022) made fundamental contributions to conversational agent alignment and showed how Proximal Policy Optimization (PPO) (Schulman et al., 2017) could be used for this purpose. PPO is however an online-RL algorithm, which carries high costs of sampling and keeping additional LLMs in memory, such as value networks.

After PPO, offline-RL algorithms emerged that are simpler and have lower computational costs. SLiC (Zhao et al., 2023) proposed a margin loss between preferred and rejected outputs, regularized with a SFT loss, RRHF (Yuan et al., 2023) extends this idea to multiple outputs. Direct Preference Optimization (DPO) (Rafailov et al., 2023) starts from the KL-controlled reward maximization objective as PPO and derives an analytical form for a reward model based on the optimal policy for this objective. Once plugged into the standard parametrization of a Bradley-Terry reward model, this yields an objective without an explicit reward and results in a contrastive gradient update rule. DPO is well funded theoretically and has found clear empirical success in aligning LLMs (Tunstall et al., 2023; Ivison et al., 2023; Jiang et al., 2024). It has also a large amount of follow-up works. Statistical Rejection Sampling Optimization (RSO) (Liu et al., 2023) proposes sampling from the optimal distribution and use a reward model for labeling, Identity Preference Optimization (IPO) (Azar et al., 2024) introduces additional regularization, Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024) derives a similar loss that does not require preference pairs and Odds Ratio Preference Optimization (ORPO) (Hong et al., 2024) enriches the SFT loss with a term based on the odds of producing the desired response.

As mentioned in Section 1, PPO and DPO originate from the same KL-controlled reward maximization which has an EBM as its optimal policy solution. Distributional Policy Gradient (DPG) (Parshakova et al., 2019) proposes using importance sampling to learn a policy matching the distribution of an EBM via KL minimization. DPG notes that, for stability purposes, offline optimization is needed. Generation with Distributional Control (GDC) (Khalifa et al., 2020a) proposes the application of DPG to controlled language generation, introduces a KL threshold to update of the offline proposal distribution and makes a connection with maximum entropy methods. GDC++ (Korbak et al., 2022) shows that, similarly to regular policy gradient, variance reduction increases performance. It also shows that distributional matching of the optimal distribution optimizes the same objective as KL-controlled reward maximization.

Similar to the above approaches, BRAIn uses distribution matching to train the policy. However, it differs from GDC and GDC++ in two major aspects. Firstly, the target distribution in BRAIn is the reward-conditioned posterior derived using Bayes' rule. Secondly, and more importantly, BRAIn uses a self-normalized baseline which results in significant variance reduction of the gradient estimate and hence, it tremendously improves performance. The self-normalized baseline yields a connection to DPO (Rafailov et al., 2023), showing that DPO-sft (a variant of DPO where the samples come from base/SFT policy and are scored using a reward model) is a special case of BRAIn.

Related to learning distributions conditioned on desired features, earlier works such as Ficler & Goldberg (2017), train special feature embeddings to produce text with desired target features. More recently Chen et al. (2021) conditions on a goodness token, and (Lu et al., 2022; Korbak et al., 2023) threshold a reward model for the same purpose. Although inspired by reward conditioning, BRAIN deviates from these approaches by not explicitly parametrizing the conditional distribution that takes both prompt x and desired reward as input. Instead, BRAIN poses the problem as distributional matching of the posterior distribution.

3. Notation

Let x be an input prompt and $\mathcal{Y}$ be the set of all output sequences. Let $p(y|x)$ be the conditional probability assigned by a supervised fine-tuned (SFT) language model (LM) to an output $y \in \mathcal{Y}$ for an input x. Let $r(x, y)$ be the corresponding reward value assigned by a given reward function. Further, let G represent a binary random variable such that the probability $p(G = 1|y, x)$ captures the goodness of a given output y for a given input x. The relationship between probability $p(G = 1|y, x)$ and reward value $r(x, y)$ depends upon the modelling assumptions made while training the reward model. In section 5.1.1, we illustrate the connection between $p(G = 1|y, x)$ and $r(x, y)$ for both absolute and relative reward models such as Bradley-Terry.

Given the prior $p(y|x)$, and the goodness model $p(G = 1|y, x)$, we define the posterior $p(y|x, G = 1)$ as our desired distribution that assigns high probability to 'good' outputs. To mimic sampling from the posterior distribution, we aim to train a new model $q_\theta(y|x)$ by minimizing the KL divergence between the two.

4. Approach

Bayesian reformulation: We first use Bayes' rule to represent the reward–conditioned posterior as:

$$p(y|x, G = 1) = \frac{p(y|x)p(G = 1|y, x)}{p(G = 1|x)} \tag{1}$$

We are interested in sampling from $p(y|x, G = 1)$, *i.e.*, samples with high reward. An obvious solution is to sample from $p(y|x)$ and keep rejecting the samples until a sample with a high reward is obtained. Despite its simplicity, rejection sampling is expensive. Hence, in this work, we propose to learn a distribution, $q_\theta(y|x)$, that can mimic sampling from the posterior without the computation overhead of rejection sampling.

Training objective: To train the parameters θ, we propose to minimize the KL–divergence between the posterior distribution $p(y|x, G = 1)$, and $q_\theta(y|x)$. This is equivalent to maximizing the objective:

$$\mathcal{L}_x(\theta) = -\mathbb{E}_{p(y|x,G=1)}\left[\log \frac{p(y|x, G = 1)}{q_\theta(y|x)}\right] \tag{2}$$

By collecting all the constant terms with respect to θ in C, the objective can be written as

$$\mathcal{L}_x(\theta) = \mathbb{E}_{p(y|x,G=1)}\left[\log q_\theta(y|x)\right] + C \tag{3}$$

Henceforth, the constant C will be omitted in subsequent formulations of the objective $\mathcal{L}_x(\theta)$.

Approximation with importance sampling: To empirically compute the expectation in eq. (3), we need to sample from the posterior $p(y|x, G = 1)$. Unfortunately, it is not possible to do so directly, and hence we resort to importance sampling (Tokdar & Kass, 2010),

$$\mathcal{L}_x(\theta) = \mathbb{E}_{q'(y|x)}\left[\frac{p(y_i|x, G = 1)}{q'(y_i|x)} \log q_\theta(y|x)\right] \tag{4}$$

where $q'(y|x)$ is an an easy–to–sample proposal distribution. Taking into account the Bayes rule in equation (1) we approximate the expectation by a sample average of n outputs $(y_1, \dots y_n)$ from $q'(y|x)$. We further use self–normalized importance sampling (ch. 9 in Owen (2013)), normalizing the weights by their sum. This results in the following loss for a given x:

$$\hat{\mathcal{L}}_x(\theta) = \sum_{i=1}^{n} \hat{\alpha}_{y_i} \log q_\theta(y_i|x), \text{ where } y_i \sim q'(y|x) \tag{5}$$

$$\hat{\alpha}_{y_i} = \frac{\alpha_{y_i}}{\sum_{j=1}^{n} \alpha_{y_j}}, \alpha_{y_i} = \frac{p(y_i|x)}{q'(y_i|x)} p(G = 1|y_i, x) \tag{6}$$

Note that we dropped $p(G = 1|x)$ from α_{y_i} as it will get cancelled due to self–normalization. We have added subscripts to α to show that they depend on the samples y. The gradient of the objective can be written as

$$\nabla_\theta \hat{\mathcal{L}}_x(\theta) = \sum_{i=1}^{n} \hat{\alpha}_{y_i} \nabla_\theta \log q_\theta(y_i|x) \tag{7}$$

Baseline subtraction: One critical issue with the loss in eq. (5) is that it assigns a positive weight to all the samples y_i for a given x, irrespective of its reward distribution $p(G = 1|y_i, x)$. In other words, the model is trained to increase the probability of all the samples and not just the high-reward ones. This is not an issue when all the samples have high reward, that is, the proposal distribution is the same as the posterior $q'(y|x) = p(y|x, G = 1)$.

When the proposal is away from the posterior, we hypothesize that it is crucial for low-reward samples to have negative weights. In GDC++ (Korbak et al., 2022), the authors proposed to subtract the following baseline from its gradient estimate:

$$Baseline = Z \frac{q_\theta(y|x)}{q'(y|x)} \nabla_\theta \log q_\theta(y|x) \tag{8}$$

where Z is the normalization constant of the target EBM (target posterior in this paper). In contrast, we propose a self-normalized baseline as described below:

To obtain a baseline for our case, we first note the following general result (derived for $q_\theta(y|x)$ here),

$$\mathbb{E}_{q_\theta(y|x)} \nabla_\theta \log q_\theta(y|x) = \mathbb{E}_{q_\theta(y|x)} \frac{1}{q_\theta(y|x)} \nabla_\theta q_\theta(y|x) \tag{9}$$

$$= \sum_{y \in \mathcal{Y}} \nabla_\theta q_\theta(y|x) = \nabla_\theta \sum_{y \in \mathcal{Y}} q_\theta(y|x) = \nabla_\theta 1 = 0 \tag{10}$$

Thus, this expectation can be subtracted from the gradient in (7) without introducing any bias. To estimate the baseline, we reuse the same samples $(y_1, \dots, y_n)$ that are used in (5) and apply self–normalized importance sampling to get:

$$\mathbb{E}_{q_\theta(y|x)} \nabla_\theta \log q_\theta(y|x) \approx \sum_{i=1}^{n} \hat{\beta}_{y_i} \nabla_\theta \log q_\theta(y_i|x) \tag{11}$$

$$y_i \sim q'(y|x) \text{ and } \hat{\beta}_{y_i} = \frac{\beta_{y_i}}{\sum_{j=1}^{n} \beta_{y_j}}, \beta_{y_i} = \frac{q_\theta(y_i|x)}{q'(y_i|x)} \tag{12}$$

Subtracting the baseline estimate from eq. (7), we get:

$$\nabla_\theta \hat{\mathcal{L}}_x(\theta) = \sum_{i=1}^{n} \left(\hat{\alpha}_{y_i} - \hat{\beta}_{y_i} \right) \nabla_\theta \log q_\theta(y_i|x) \tag{13}$$

$y_i, \hat{\alpha}_{y_i}$, and $\hat{\beta}_{y_i}$ are same as in eqs. (6) and (12). We call the self-normalized baseline subtracted gradient estimate in (13) the **BRAIN gradient estimate**. Intuitively, α_{y_i} and β_{y_i} are proportional to the posterior $p(y_i|x, G = 1)$ and policy $q_\theta(y_i|x)$ distributions, respectively. Thus, the weight (difference of normalized α_{y_i} and β_{y_i}) of each y_i captures how far the current estimate of policy $q_\theta(y_i|x)$ is from the true posterior $p(y_i|x, G = 1)$. This also alleviates the critical issue of assigning positive weights to all y_i irrespective of their reward. Samples with lower reward, and hence lower posterior $p(y_i|x, G = 1)$ (consequently lower α_{y_i}), will get negative weights as soon as the policy distribution $q_\theta(y_i|x)$ assigns higher probability to them (consequently higher β_{y_i}) than the posterior, and vice-versa.

Algorithm 1 Bayesian Reward-conditioned Amortized Inference

```
Require: $r(x, y), p(y|x), D, e, m, n, k$
 1: Initialize $q'(y|x) \leftarrow p(y|x)$, $q_\theta(y|x) \leftarrow p(y|x)$
 2: Initialize $D_0 \leftarrow \{\}$
 3: for $t = 1$ to $e$ do
 4:     $S_t \leftarrow$ randomly selected $m$ prompts from $D$
 5:     for all $x \in S_t$ do
 6:         $Y_x \leftarrow$ generate $n$ samples using $q'(y|x)$
 7:         for all $y \in Y_x$ do
 8:             Cache $\log q'(y|x), \log p(y|x), r(x, y)$ in $S_t$
 9:             Cache normalized $\hat{\alpha}_y$ (eq. (6) or eq. (20)) in $S_t$
10:         end for
11:     end for
12:     $D_t \leftarrow D_{t-1} \cup S_t$
13:     for $j = 1$ to $k$ do
14:         $B \leftarrow$ randomly sampled batch from $D_t$
15:         for all $(x, Y_x, q'(y|x), \hat{\alpha}_y) \in B$ do
16:             Compute $\hat{\beta}_y$ (eq. (12))
17:             $\mathcal{L}_x(\theta) \leftarrow \sum_{y \in Y_x} (\hat{\alpha}_y - \hat{\beta}_y) \log q_\theta(y|x)$
18:         end for
19:         Update $\theta$ using $\nabla_\theta \sum_{x \in B} \mathcal{L}_x(\theta)$
20:     end for
21:     $q'(y|x) \leftarrow q_\theta(y|x)$
22: end for
23: Return $q_\theta$
```

The BRAIN algorithm with the self-normalized baseline is given in Algorithm 1. We initialize both our proposal $q'(y|x)$ and policy $q_\theta(y|x)$ with $p(y|x)$. We update the proposal e times after every k gradient updates. m is the number of prompts sampled from dataset D after updating the proposal, and n is the number of outputs generated for each prompt $x \in D$.

In our experiments, we show that subtracting the self–normalized baseline results in a large improvement in performance. We hypothesize that the baseline allows the model to focus on distinctive features of high-reward outputs compared to the lower-reward ones. A formal justification of the resultant BRAIN gradient estimate is provided in the Appendix 4.1.

4.1. A formal justification of the BRAIN gradient estimate:

Self-normalized importance sampling (SNIS) introduces bias in any estimator. In this paper, we have used SNIS to approximate the importance weights as well as the baseline in our gradient estimate. Using biased gradient estimators for training often results in optimization of an objective different from the desired one.

However, in our case, we show that using the BRAIN gradient estimate for training, results in minimizing a self-normalized version of KL–divergence (defined below) whose minimum value is achieved only when $q_\theta(y|x) = p(y|x, G = 1)$. Note that this is a consequence of the chosen self-normalized baseline in (12).

First, we define self-normalized KL divergence in the context of this paper. Next, we show that our proposed BRAIN gradient estimate is an unbiased estimator of the gradient of this divergence measure. Finally, we prove that this self-normalized KL divergence is non-negative and equals 0 only when the policy learns to mimic the posterior. The proofs are in appendix A.

Definition 4.1. Let the proposal distribution $q'(y|x)$, training policy $q_\theta(y|x)$ and posterior $p(y|x, G = 1)$ be as defined earlier. Furthermore, we assume that support$(p) \subseteq$ support$(q_\theta) \subseteq$ support(q'). For any n outputs, $Y_x = (y_1, \ldots, y_n)$, let $\hat{\alpha}_{y_i}$ and $\hat{\beta}_{y_i}$ be the self–normalized importance sampling weights for the loss and the baseline, respectively (eqs. (6) and (12)). The self-normalized KL–divergence between the posterior and training policy for the given proposal distribution is defined as:

$$\begin{aligned} &D_n^{q'}(p(y|x,G=1)||q_\theta(y|x)) \\ &= \mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \left[D_{KL}(\hat{\alpha}_{Y_x} || \hat{\beta}_{Y_x}) \right] \text{ where} \end{aligned} \tag{14}$$

$$D_{KL}(\hat{\alpha}_{Y_x} || \hat{\beta}_{Y_x}) = \sum_{i=1}^{n} \hat{\alpha}_{y_i} \log \frac{\hat{\alpha}_{y_i}}{\hat{\beta}_{y_i}} \tag{15}$$

Theorem 4.2. *The* BRAIN *gradient estimate defined in* (13) *is an unbiased estimator of the gradient (w.r.t. θ) of negative self-normalized KL–divergence between the posterior $p(y|x, G = 1)$ and training policy $q_\theta(y|x)$ defined in* (14). *Here, the dependence of KL divergence on θ comes from $\hat{\beta}_{y_i}$ being a function of $q_\theta(y_i|x)$.*

Since the gradient of our objective in eq. (13) is the same as the gradient of negative self-normalized KL-divergence defined in eq. (14), in the rest of the paper, we refer to negative self-normalized KL-divergence between the posterior and the training policy as **BRAIN objective**.

Theorem 4.3. *The self-normalized KL–divergence defined in* (14) *reaches its minimum value of* 0 *if and only if the KL–divergence between the posterior and training policy also reaches* 0. *Also, $q_\theta(y|x) = p(y|x, G = 1)$ is the only minima of self-normalized KL–divergence defined in* (14).

5. Connection with existing RLHF methods

In this section, we show that the proposed BRAIN algorithm acts as a bridge between two disparate objectives used for reinforcement learning in LLMs as shown in Figure 1.

1. Distribution matching objectives as described in (Korbak et al., 2022; Parshakova et al., 2019; Khalifa et al., 2020b).
2. The contrastive-training approach presented in DPO (Rafailov et al., 2023), specifically DPO-sft where the samples come from the base/SFT policy and are scored using the reward model.

In the rest of the section:

1. We establish the connection between the target distribution of BRAIN as defined in (1), and the PPO-optimal policy which is the target distribution for PPO, DPO as well as distribution matching methods for RLHF. Specifically, we establish that under Bradley-Terry preference modelling assumptions, the posterior becomes equal to the PPO-optimal policy.
2. We derive the DPO-sft gradient estimate as a special case of the BRAIN gradient estimate.

5.1. Posterior and PPO-optimal policy

In this section, we show that for a Bradley-Terry preference model, the posterior $p(y|x, G = 1)$ corresponds to the PPO-optimal policy. Before we jump into the proof, we briefly describe Bradley-Terry preference model and its connection with the goodness model in BRAIN $p(G = 1|y, x)$.

5.1.1. BRADLEY-TERRY PREFERENCE MODEL FOR LLMS

Given an absolute goodness measure g_i for each y_i, Bradley-Terry Model (BTM) defines the probability of choosing y_i over y_j as:

$$\mathbb{P}(y_i \succ y_j) = \frac{g_i}{g_i + g_j} \tag{16}$$

Recall that in our formulation, the binary random variable G represents the goodness of an output, and hence, the conditional probability $p(G = 1|y_i, x)$ can be used as a proxy for g_i:

$$\mathbb{P}(y_i \succ y_j|x) = \frac{p(G=1|y_i,x)}{p(G=1|y_i,x) + p(G=1|y_j,x)} \tag{17}$$

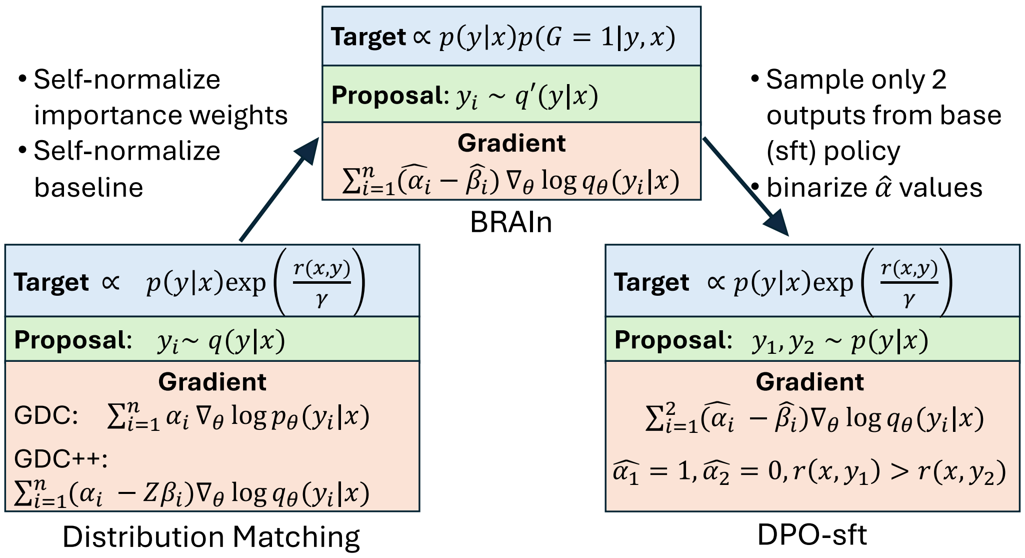
Target $\propto p(y|x)p(G=1|y,x)$
Proposal: $y_i \sim q'(y|x)$
Gradient
$\sum_{i=1}^{n}(\widehat{\alpha_i} - \widehat{\beta_i})\nabla_\theta \log q_\theta(y_i|x)$
BRAIn
• Self-normalize importance weights
• Self-normalize baseline
• Sample only 2 outputs from base (sft) policy
• binarize $\hat{\alpha}$ values
Target $\propto p(y|x)\exp\left(\frac{r(x,y)}{\gamma}\right)$
Proposal: $y_i \sim q(y|x)$
Gradient
GDC: $\sum_{i=1}^{n}\alpha_i \nabla_\theta \log p_\theta(y_i|x)$
GDC++: $\sum_{i=1}^{n}(\alpha_i - Z\beta_i)\nabla_\theta \log q_\theta(y_i|x)$
Distribution Matching
Target $\propto p(y|x)\exp\left(\frac{r(x,y)}{\gamma}\right)$
Proposal: $y_1, y_2 \sim p(y|x)$
Gradient
$\sum_{i=1}^{2}(\widehat{\alpha_i} - \widehat{\beta_i})\nabla_\theta \log q_\theta(y_i|x)$
$\widehat{\alpha_1} = 1, \widehat{\alpha_2} = 0, r(x,y_1) > r(x,y_2)$
DPO-sft

Figure 1. BRAIN acts as a bridge between distribution matching methods (GDC (Khalifa et al., 2020b) and GDC++ (Korbak et al., 2022)) and DPO (Rafailov et al., 2023), specifically DPO-sft where the samples come from the base/SFT policy. The values α_i, $\hat{\alpha}_i$ and $\hat{\beta}_i$ are as defined in equations (6) and (12) whereas Z is the normalization constant of the target. Note that the proposal distribution in the distribution matching methods and BRAIN is chosen differently.

During training of the reward function $r(x, y)$, we are given triplets (x, y_i, y_j) such that for an input x, the response y_i is preferred over the response y_j. We train it by maximizing the log-likelihood over all the training triplets. During training, the goodness measure g_i is parameterized as $\exp\left(\frac{r(x,y_i)}{\gamma}\right)$ resulting in the following log-likelihood of a triplet:

$$\log \mathbb{P}(y_i \succ y_j|x) = \log \sigma\left(\frac{r(x,y_i) - r(x,y_j)}{\gamma}\right) \tag{18}$$

Thus, by equating eq. (18) and log of eq. (17), we get the maximum likelihood estimate of the log–odds as:

$$\log \frac{p(G=1|y_i,x)}{p(G=1|y_j,x)} = \frac{r(x,y_i) - r(x,y_j)}{\gamma} \tag{19}$$

The above ratio is exactly what we need to compute self–normalized importance weights $\alpha_{y_i} / \sum_{j=1}^{n} \alpha_{y_j}$ in eq. (5). We formalize this result in the proposition below.

Proposition 5.1. *For the Bradley-Terry preference model defined in eq. (17) and parameterized by eq. (18), the self–normalized importance-weights $\hat{\alpha}_{y_i} = \alpha_{y_i} / \sum_{j=1}^{n} \alpha_{y_j}$ have the following form:*

$$\frac{\exp(\frac{r(x,y_i)}{\gamma} + \log p(y_i|x) - \log q'(y_i|x))}{\sum_{j=1}^{n} \exp(\frac{r(x,y_j)}{\gamma} + \log p(y_j|x) - \log q'(y_j|x))} \tag{20}$$

In the special case where the proposal distribution $q'(y|x)$ is the same as the prior distribution $p(y|x)$, the importance weights reduces to

$$\hat{\alpha}_{y_i} = \frac{\exp\left(\frac{r(x,y_i)}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(\frac{r(x,y_j)}{\gamma}\right)} \tag{21}$$

See the appendix for the proof. Observe that $\hat{\alpha}_{y_i}$ in eq. (21) is nothing but a softmax over the reward values $r(x, y_i)$. In the next section, we show that DPO is a special case of our formulation when we replace softmax with argmax and set $n = 2$. This amounts to assigning all the importance weight to the output sample with the most reward.

Theorem 5.2. *For a Bradley-Terry reward model, the posterior $p(y|x, G = 1)$ is same as the PPO optimal policy given by:*

$$p^*(y|x) = \frac{p(y|x)\exp(\frac{r(x,y)}{\gamma})}{\left(\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)\exp\left(\frac{r(x,\bar{y})}{\gamma}\right)\right)} \tag{22}$$

The above eqn. for PPO optimal policy is as shown in equation(4) in Rafailov et al. (2023). Note that the reference policy in DPO is same as the prior policy in our formulation.

5.2. DPO-sft as a special case of BRAIN

In DPO (Rafailov et al., 2023), an ideal setting is discussed where the samples are generated from the base policy ($p(y|x)$ in our case) and annotated by humans. Often, the preference pairs in publicly available data are sampled from a different policy than $p(y|x)$. To address this, Liu et al. (2023) experiment with a variant of DPO, called DPO-sft, in which a reward model is first trained on publicly available preference pairs and then used to annotate the samples generated from the base policy $p(y|x)$.

We claim that BRAIN reduces to DPO-sft when we generate only 2 samples per input x and assign the importance weight of the winner to 1. The last assumption is equivalent to replacing softmax in eq. (21) with an argmax. We formalize it in the following theorem:

Theorem 5.3. *Let the proposal distribution $q'(y|x)$ in* BRAIN *be restricted to the prior $p(y|x)$. When $n = 2$, and the softmax is replaced by argmax in eq. (21), then the* BRAIN *objective reduces to DPO objective proposed by Rafailov et al.*

Proof. We start with BRAIN objective defined in eq. (14) and insert our assumptions to arrive at DPO objective.

By substituting $n = 2$ and $q'(y|x) = p(y|x)$ in R.H.S. of eq. (14), and using eq. (15), we get:

$$\mathbb{E}_{y_1,y_2\sim p(y|x)}\left[\hat{\alpha}_{y_1}\log\frac{\hat{\alpha}_{y_1}}{\hat{\beta}_{y_1}} + \hat{\alpha}_{y_2}\log\frac{\hat{\alpha}_{y_2}}{\hat{\beta}_{y_2}}\right] \tag{23}$$

Now, without loss of generality, assume that $r(x, y_1) > r(x, y_2)$. Replacing softmax with argmax in eq. (21), we get $\hat{\alpha}_{y_1} = 1$ and $\hat{\alpha}_{y_2} = 0$. Plug it in eq. (23) to get:

$$-\mathbb{E}_{y_1,y_2\sim p(y|x)}\log\hat{\beta}_{y_1}$$

Now recall from eq. (12) that $\hat{\beta}_{y_1} = \beta_{y_1}/(\beta_{y_1} + \beta_{y_2})$ and $\beta_{y_i} = \frac{q_\theta(y_i|x)}{p(y_i|x)}$. Replacing it in the above equation and rearranging the terms, we get:

$$-\mathbb{E}_{y_1,y_2\sim p(y|x)}\log\sigma\left(\log\frac{q_\theta(y_1|x)}{p(y_1|x)} - \log\frac{q_\theta(y_2|x)}{p(y_2|x)}\right)$$

The above expression is exactly same as equation (7) in Rafailov et al..

□

We also note that to extend DPO to multiple outputs, *i.e.*, $n > 2$, Rafailov et al. resorts to a more general Plackett-Luce preference model.

6. Experimental Setup

Tasks: We conduct experiments on two natural language generation tasks, *viz.*, summarization, and multi-turn dialog. In the summarization task, we align CarperAI's summarization LM to a Bradley-Terry reward model. We train and evaluate using Reddit TL;DR dataset, in which input is a post on Reddit, and the aligned model should generate a high reward summary.

Table 1. Win–rate (in %age) ± 95% confidence–interval against the gold output on the test sets. Train RM corresponds to the reward model used during training, whereas for LLM eval., we prompt Mixtral-8x7B.

	DPO	**DPO-sft**	**RSO**	**BRAIN**
AnthropicHH				
Train RM	54.60±1.37	87.37±0.91	84.59±0.99	95.40±0.57
LLM eval	67.02±1.29	66.68±1.29	67.22±1.29	74.36±1.20
Reddit TL;DR				
Train RM	86.72±0.82	90.86±0.70	91.24±0.68	95.21±0.52
LLM eval	60.26±1.18	60.55±1.18	60.41±1.18	64.74±1.16

In the multi-turn dialog task, we ensure that a open-ended chatbot's responses are **H**elpful and **H**armless. We use QLoRA-tuned Llama-7b (Dettmers et al., 2023) as SFT model, and a fine-tuned GPT-J (Wang & Komatsuzaki, 2021) as the reward model. We train and evaluate using a subset of AntrophicHH (Bai et al., 2022) dataset. See appendix B.1 for hyperlinks to all the datasets and models described above.

Evaluation metrics: We use win–rate over gold responses and direct win–rate against the baseline responses to measure the performance of various techniques. Win–rate is defined as the fraction of test samples on which the generated response gets a higher reward than the gold response. We use two independent reward functions to compute the win–rate: (1) *Train RM*, which is the reward function used to align the SFT model, and (2) *LLM eval*, which follows LLM-as-a-judge (Zheng et al., 2023) and prompts a strong instruction following LLM, Mixtral-8x7B (Jiang et al., 2024), to compare the two outputs and declare a winner. The first metric captures the effectiveness of the alignment method in maximizing the specified reward, while a high win–rate using the LLM prompting ensures that the alignment method is not resorting to reward-hacking (Skalse et al., 2022).

We prompt GPT-4 (OpenAI et al., 2023) to directly compare BRAIN's responses with each of the baselines separately. See appendix B.3 for the specific instructions used for prompting GPT-4 and Mixtral-8x7B.

Training details: While DPO uses human-annotated data directly, we generate $n = 32$ samples per input prompt for the other models (DPO-sft, RSO, and BRAIN). The samples are organized in 16 pairs for DPO-sft and RSO, while for BRAIN, the 32 samples are grouped together. We use $\gamma = 1$ for BRAIN in all our experiments, unless otherwise stated, and average the log-probability of the tokens. We train each model for a total of $10,000$ steps ($e = 40$ and $k = 250$ in algorithm 1) with 4 prompts and 32 outputs per prompt in a batch B. The model is evaluated after every $1,000$ steps and the best model is selected based on the win rate against the gold response on the cross-validation set. Other training

Table 2. Head-to-head comparison of the BRAIN against the baselines using GPT-4.

	AnthropicHH			Reddit TL;DR		
	Win %	Tie %	Loss %	Win %	Tie %	Loss %
vs DPO	44.0	31.6	21.4	42.1	19.6	38.3
vs DPO-sft	45.4	34.4	20.2	45.2	18.9	35.9
vs RSO	44.2	35.5	20.3	45.1	17.6	37.3

details are given in Appendix B.2.

7. Experimental Results

Research Questions: Our experiments aim to assess BRAIN's performance in aligning an SFT model to a given reward function and compare it against various baselines. Specifically, we answer the following research questions:

1. How does BRAIN compare to existing baselines?
2. How does the KL-reward frontier of BRAIN compare with that of DPO?
3. What is the role of self-normalized baseline subtraction in BRAIN?
4. How to bridge the gap between DPO and BRAIN?
5. How does varying the number of output samples per input affect BRAIN's performance?

7.1. Comparison with baselines

Table 1 compares the win–rate of BRAIN against our baselines – RSO, DPO, and DPO-sft. We observe that BRAIN consistently outperforms all the baselines, irrespective of the model used to compute win–rates. Specifically, when measured using the specified reward model, BRAIN achieves 8 and 4 pts better win–rate than the strongest baseline on AntrophicHH and TL;DR, respectively. Next, we observe that even though DPO-sft has a much better win–rate than DPO when computed using the specified reward function, their performance is the same when judged by an independent LLM, indicative of reward–hacking.

In Table 2, we summarize the results of head-to-head comparisons of BRAIN with each of the baselines judged by GPT-4 on a set of 500 examples from each of the dataset. *Win %* indicates the percentage of examples for which BRAIN was declared the winner compared to the baseline. We observe that BRAIN wins twice as many times as the baselines on AnthropicHH.

7.2. KL-reward frontier

Next, we compare the KL-reward frontier of BRAIN with that of DPO-sft by varying the value of γ (β in the case of DPO-sft) and selecting multiple checkpoints for each β/γ. For each checkpoint, we sample 1000 prompts and generate 8 outputs per prompt. For each output, we compute

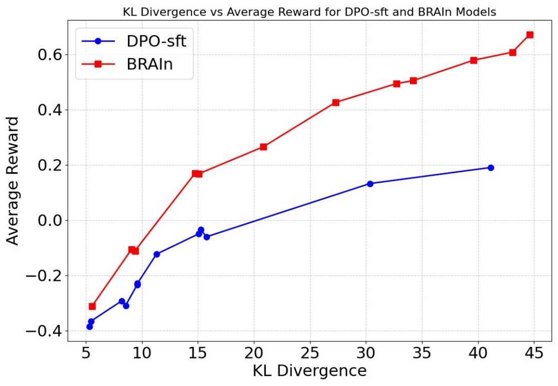

Figure 2. The KL-reward frontier of BRAIN and DPO-sft

the reward $r(x, y)$ and $\log q_\theta(y|x) - \log p(y|x)$ and average them over the outputs and the prompts. This gives us $(KL, reward)$ for each checkpoint. We plot these values in Figure 2.

As can be observed from the plot, for the same KL divergence, BRAIN achieves a much higher reward than DPO-sft. Moreover, DPO-sft fails to achieve a reward as high as BRAIN no matter how much the beta value is decreased.

7.3. Role played by self-normalized baseline

Next, we demonstrate the crucial role that the self-normalized baseline plays in BRAIN.

Our first experiment is a toy experiment where the posterior/-target is defined to be the 1-D standard Gaussian distribution $\mathcal{N}(0, 1)$. The training policy q_θ is also Gaussian $\mathcal{N}(1, 1)$. We generate samples from the proposal distribution which is also chosen to be Gaussian $\mathcal{N}(\theta, 1)$ where θ is varied from 0 to 1. For each value of θ, we generate 8 samples from the proposal distribution to compute the GDC, GDC++ (Korbak et al., 2022; Khalifa et al., 2020b), and BRAIN gradient estimate. We repeat this experiment 2000 times and compute the variance across the different sets. The results are plotted in Figure 3.

	BRAIn	w/o self-norm	w/o baseline
TL;DR	95.2	61.4	61.1
AnthropicHH	95.4	59.1	58.3

Table 3. Effect of self-normalized baseline on the performance of various models.

Next, we demonstrate the effect of self-normalized baseline on the performance of BRAIN on TL;DR and Anthropic datasets. Table 3 summarizes our observations about the role

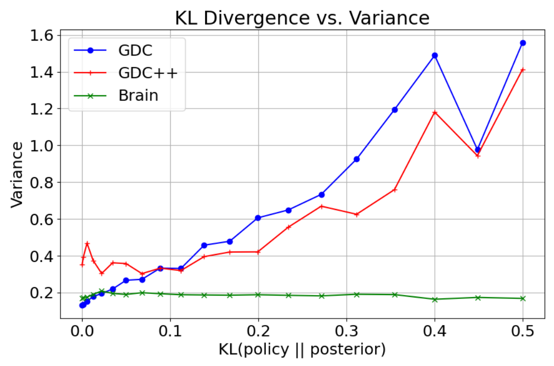

Figure 3. Variance of GDC, GDC++ and BRAIN gradient estimates

Table 4. Win–rates (in %age) obtained by incrementally augmenting DPO-sft; DPO-sft+IW: replace argmax by softmax; DPO-sft+IW+n: take softmax over $n = 32$ outputs, instead of two outputs in 16 pairs

	DPO-sft	**DPO-sft +IW**	**DPO-sft +IW+n**	**BRAIN**
Train RM	87.37±0.91	89.21±0.85	93.30±0.69	95.40±0.57
LLM eval	66.78±1.29	69.28±1.27	73.89±1.21	74.36±1.17

that self-normalization of baseline plays in performance. As can be observed, there is a drastic reduction in performance if self-normalization in the baseline is removed.

7.4. Bridging the gap between DPO-sft and BRAIN

In section 5.2, we show that under certain restrictions, BRAIN objective reduces to DPO-sft objective. In this section, we start with DPO-sft and relax these restrictions one at a time to demonstrate the impact of each restriction. First, we get rid of argmax and reintroduce softmax over rewards (eq. (21)) to compute self–normalized importance weights $\hat{\alpha}_{y_i}$. This corresponds to the objective in eq. (23). We call this DPO-sft+IW. Next, we relax the assumption of $n = 2$, and take softmax over $n = 32$ outputs instead of softmax over two samples in 16 pairs. We call it DPO-sft+IW+n. Finally, we relax the restriction of proposal distribution to be the prior distribution only and instead update our proposal periodically (algorithm 1) to arrive at BRAIN.

Table 4 compares the win–rate of the two intermediate models (DPO-sft+IW, and DPO-sft+IW+n) with DPO-sft and BRAIN over the AntropicHH dataset. We first observe that relaxing each restriction consistently improves both the win–rates. The biggest gain ($\sim$ 4 pts) comes by relaxing the assumption of $n = 2$ and taking a softmax over all 32 outputs. This is expected as information contained in simultaneously comparing 32 outputs is potentially more than only 16 pairs.

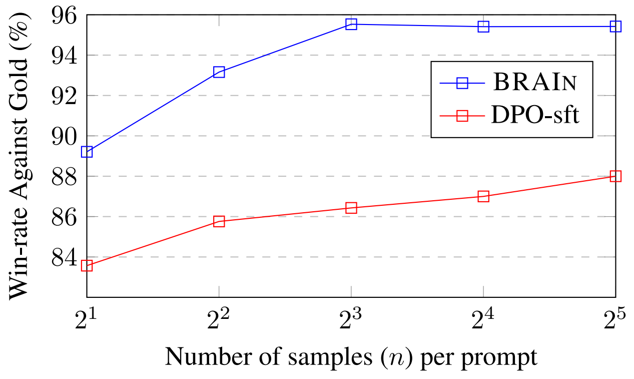

Figure 4. Plot of Win-rate Against Gold as a function of the Number of Samples per Prompt.

7.5. Effect of the number of output samples

Next, we study the effect of varying the number of output samples (n) per input prompt x on the performance of BRAIN. We retrain DPO-sft and BRAIN on AntropicHH dataset for each $n \in \{2, 4, 8, 16, 32\}$. As done earlier, we create $n/2$ pairs from the n samples while training DPO-sft. The win–rates computed by specified reward model (Train RM) for each n are plotted in section 7.5. We observe that including more samples in BRAIN objective leads to improvement in performance till $n = 8$ after which it saturates, whereas the performance of DPO-sftimproves monotonically, albeit slowly.

8. Conclusion

In this paper, we propose an LLM alignment algorithm called BRAIN: **B**ayesian **R**eward-conditioned **A**mortized **In**ference. The primary novelty in BRAIN is the inclusion of a self-normalized baseline in the gradient estimate of distribution matching objectives for RL, that we refer to as BRAIN gradient estimate. Theoretically, the self-normalized baseline helps us to establish a connection between distribution matching methods and DPO, showing that DPO (DPO-sft, to be specific) is a special case of BRAIN. We further establish that the BRAIN gradient estimate is the gradient of a self-normalized version of KL divergence whose properties we intend to explore in future work. Additionally, we generalize the target distribution in PPO/DPO using Bayes' rule. The experimental results demonstrate the superiority of BRAIN over other RLHF methods.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Azar, M. G., Guo, Z. D., Piot, B., Munos, R., Rowland, M., Valko, M., and Calandriello, D. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Bai, Y., Jones, A., and et al., K. N. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

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

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Ficler, J. and Goldberg, Y. Controlling linguistic style aspects in neural language generation. In Brooke, J., Solorio, T., and Koppel, M. (eds.), *Proceedings of the Workshop on Stylistic Variation*, pp. 94–104, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4912. URL https://aclanthology.org/W17-4912.

Hong, J., Lee, N., and Thorne, J. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*, 2024.

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.

Ivison, H., Wang, Y., Pyatkin, V., Lambert, N., Peters, M., Dasigi, P., Jang, J., Wadden, D., Smith, N. A., Beltagy, I., et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.

Jaques, N., Gu, S., Turner, R. E., and Eck, D. Tuning recurrent neural networks with reinforcement learning. 2017.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de Las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts. *CoRR*, abs/2401.04088, 2024. doi: 10.48550/ARXIV.2401.04088. URL https://doi.org/10.48550/arXiv.2401.04088.

Khalifa, M., Elsahar, H., and Dymetman, M. A distributional approach to controlled text generation. *arXiv preprint arXiv:2012.11635*, 2020a. URL https://arxiv.org/pdf/2012.11635.

Khalifa, M., Elsahar, H., and Dymetman, M. A distributional approach to controlled text generation. In *International Conference on Learning Representations*, 2020b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Korbak, T., Elsahar, H., Kruszewski, G., and Dymetman, M. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *Advances in Neural Information Processing Systems*, 35:16203–16220, 2022.

Korbak, T., Shi, K., Chen, A., Bhalerao, R. V., Buckley, C., Phang, J., Bowman, S. R., and Perez, E. Pretraining language models with human preferences. In *International Conference on Machine Learning*, pp. 17506–17533. PMLR, 2023.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J., and Liu, J. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.

Lu, X., Welleck, S., Hessel, J., Jiang, L., Qin, L., West, P., Ammanabrolu, P., and Choi, Y. Quark: Controllable text generation with reinforced unlearning. *Advances in neural information processing systems*, 35:27591–27609, 2022.

OpenAI, :, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., et al. GPT-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Owen, A. B. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

Parshakova, T., Andreoli, J., and Dymetman, M. Distributional reinforcement learning for energy-based sequential models. *CoRR*, abs/1912.08517, 2019. URL http://arxiv.org/abs/1912.08517.

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.

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

Skalse, J., Howe, N. H. R., Krasheninnikov, D., and Krueger, D. Defining and characterizing reward hacking. *CoRR*, abs/2209.13085, 2022. doi: 10.48550/ARXIV.2209.13085. URL https://doi.org/10.48550/arXiv.2209.13085.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize from human feedback. *CoRR*, abs/2009.01325, 2020. URL https://arxiv.org/abs/2009.01325.

Tokdar, S. T. and Kass, R. E. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

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

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

Völske, M., Potthast, M., Syed, S., and Stein, B. Tl;dr: Mining reddit to learn automatic summarization. In Wang, L., Cheung, J. C. K., Carenini, G., and Liu, F. (eds.), *Proceedings of the Workshop on New Frontiers in Summarization, NFiS@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pp. 59–63. Association for Computational Linguistics, 2017. doi: 10.18653/V1/W17-4508. URL https://doi.org/10.18653/v1/w17-4508.

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

Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., Zhou, J., Chen, S., Gui, T., Zhang, Q., and Huang, X. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models, 2023.

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.

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

Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685, 2023. doi: 10.48550/ARXIV.2306.05685. URL https://doi.org/10.48550/arXiv.2306.05685.

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

A. Proof of Theorems

For the sake of completeness, we restate the definitions and the theorems here.

Definition 4.1. Let the proposal distribution $q'(y|x)$, training policy $q_\theta(y|x)$ and posterior $p(y|x, G = 1)$ be as defined earlier. Furthermore, we assume that support$(p) \subseteq$ support$(q_\theta) \subseteq$ support(q'). For any n outputs, $Y_x = (y_1, \dots, y_n)$, let $\hat{\alpha}_{y_i}$ and $\hat{\beta}_{y_i}$ be the self–normalized importance sampling weights for the loss and the baseline, respectively (eqs. (6) and (12)). The self-normalized KL–divergence between the posterior and training policy for the given proposal distribution is defined as:

$$\begin{aligned} & D_n^{q'}(p(y|x, G = 1)||q_\theta(y|x)) \\ & = \mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \left[D_{KL}(\hat{\alpha}_{Y_x}||\hat{\beta}_{Y_x}) \right] \text{ where} \qquad (14) \\ & D_{KL}(\hat{\alpha}_{Y_x}||\hat{\beta}_{Y_x}) = \sum_{i=1}^{n} \hat{\alpha}_{y_i} \log \frac{\hat{\alpha}_{y_i}}{\hat{\beta}_{y_i}} \qquad (15) \end{aligned}$$

Theorem 4.2. *The* BRAIN *gradient estimate defined in* (13) *is an unbiased estimator of the gradient (w.r.t. θ) of negative self-normalized KL–divergence between the posterior $p(y|x, G = 1)$ and training policy $q_\theta(y|x)$ defined in* (14)*. Here, the dependence of KL divergence on θ comes from $\hat{\beta}_{y_i}$ being a function of $q_\theta(y_i|x)$.*

Proof. For unbiasedness of the BRAIN gradient estimator, we need to show the following

$$\mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \sum_{i=1}^{n} \left(\hat{\alpha}_{y_i} - \hat{\beta}_{y_i} \right) \nabla_\theta \log q_\theta(y_i|x) = -\nabla_\theta D_n^{q'}(p(y|x, G = 1)||q_\theta(y|x)), \qquad (24)$$

where $\mathbf{y} = (y_1, \dots, y_n)$ is a sequence of n outputs, the values of α and β are as defined in (6) and (12) respectively. The superscripts in α and β denote their explicit dependence on the output sequence.

To prove the above, we expand each negative KL–divergence term in the self-normalized KL–divergence in terms of the entropy of normalized α_{y_i} and the cross-entropy between self-normalized α_{y_i} and β_{y_i}.

$$-D_n^{q'} = \mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \left[\mathcal{H}\left(\hat{\alpha}_{y_i}\right) + \sum_{i=1}^{n} \hat{\alpha}_{y_i} \log \hat{\beta}_{y_i} \right] \qquad (25)$$

Since the values α don't depend on θ, they are discarded during the gradient computation. Hence, the gradient can be written as

$$-\nabla_\theta D_n^{q'} = \mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \sum_{i=1}^{n} \hat{\alpha}_{y_i} \nabla_\theta \left[\log \hat{\beta}_{y_i} \right] \qquad (26)$$

Noting that $\hat{\beta}_{y_i} = \frac{\beta_{y_i}}{\sum_{j=1}^{n} \beta_{y_j}}$, we split the logarithm and compute the gradient of each term separately.

$$\begin{aligned} -\nabla_\theta D_n^{q'} &= \mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \sum_{i=1}^{n} \hat{\alpha}_{y_i} \left[\nabla_\theta \log \beta_{y_i} - \nabla_\theta \log \sum_{j=1}^{n} \beta_{y_j} \right] \qquad (27) \\ &= \mathbb{E}_{\substack{y_i \sim q'(y|x) \\ 1 \le i \le n}} \sum_{i=1}^{n} \hat{\alpha}_{y_i} \left[\nabla_\theta \log \beta_{y_i} - \frac{\sum_{j=1}^{n} \nabla_\theta \beta_{y_j}}{\sum_{j=1}^{n} \beta_{y_j}} \right] \qquad (28) \end{aligned}$$

Next, we note that $\nabla_\theta \beta_{y_i} = \beta_{y_i} \nabla_\theta \log \beta_{y_i}$ and

$$\nabla_\theta \log \beta_{y_i} = \nabla_\theta \left[\log q_\theta(y_i|x) - \log q'(y_i|x) \right] = \beta_{y_i} \nabla_\theta \log q_\theta(y_i|x) \qquad (29)$$

Replacing these results back in equation (28), we get the desired result.

$$-\nabla_\theta D_n^{q'} = \mathbb{E}_{\substack{y_i \sim q'(y|x)\\ 1\le i\le n}} \sum_{i=1}^{n} \hat{\alpha}_{y_i} \left[\nabla_\theta \log q_\theta(y_i|x) - \frac{\sum_{j=1}^{n} \beta_{y_j} \nabla_\theta \log q_\theta(y_j|x)}{\sum_{j=1}^{n} \beta_{y_j}} \right] \tag{30}$$

$$= \mathbb{E}_{\substack{y_i \sim q'(y|x)\\ 1\le i\le n}} \left[\frac{\sum_{i=1}^{n} \alpha_{y_i} \nabla_\theta \log q_\theta(y_i|x)}{\sum_{j=1}^{n} \alpha_{y_j}} - \frac{\sum_{i=1}^{n} \beta_{y_i} \nabla_\theta \log q_\theta(y_i|x)}{\sum_{j=1}^{n} \beta_{y_j}} \right] \tag{31}$$

$$= \mathbb{E}_{\substack{y_i \sim q'(y|x)\\ 1\le i\le n}} \sum_{i=1}^{n} \left[\hat{\alpha}_{y_i} - \hat{\beta}_{y_i} \right] \nabla_\theta \log q_\theta(y_i|x) \tag{32}$$

□

Theorem 4.3. *The self-normalized KL–divergence defined in* (14) *reaches its minimum value of* 0 *if and only if the KL–divergence between the posterior and training policy also reaches* 0*. Also,* $q_\theta(y|x) = p(y|x, G = 1)$ *is the only minima of self-normalized KL–divergence defined in* (14)*.*

Proof. First, we will prove that $D_{KL} = 0 \implies D_n^{q'} = 0$ which is its minimum value. We note that the self-normalized KL–divergence is the weighted sum of KL–divergence between the normalized values of α and β. Hence, from the property of KL–divergence, it can't be negative, that is, $D_n^{q'} \ge 0$. Now, lets assume that $D_{KL}(p(y|x, G = 1)||q_\theta(y|x)) = 0$. By the property of KL–divergence, this implies that $p(y|x, G = 1) = q_\theta(y|x) \forall y \in \mathcal{Y}$. Thus:

$$\beta_{y_i} = \frac{q_\theta(y_i|x)}{q'(y_i|x)} = \frac{p(y_i|x, G = 1)}{q'(y_i|x)} = \alpha_{y_i} \tag{33}$$

Incorporating it in the definition of self-normalized KL divergence in (14), we get:

$$D_n^{q'}(p(y|x, G = 1)||q_\theta(y|x)) = \mathbb{E}_{\mathbf{y} \sim q'(y|x)} \left[D_{KL}\left(\hat{\alpha}_{y_i} \,||\, \hat{\alpha}_{y_i} \right) \right] = 0 \tag{34}$$

Instead of proving the other direction, we provide a constructive proof of its contrapositive, that is, $D_{KL} \neq 0 \implies D_n^{q'} \neq 0$. To see this, we note that $D_{KL} \neq 0$ implies the existence of at least one output, say y_+, where the posterior and policy disagree. Without loss of generality, lets assume that $p(y_+|x, G = 1) > q_\theta(y_+|x)$. Since the probabilities must up to 1, there must exist at least one output, say y_-, for which $p(y_-|x, G = 1) < q_\theta(y_-|x)$.

Since self-normalized KL-divergence $D_n^{q'}$ has a KL-divergence D_{KL} term for every sequence of length n, we construct a sequence $\mathbf{y} = (y_+, y_-, \dots, y_-)$. For such a sequence, all the values of α except the first one are equal. We note that the importance weight of the first output, that is $\alpha_{y_+} = \frac{p(y_+|x,G=1)}{q'(y_+|x)} > \frac{q_\theta(y_+|x)}{q'(y_+|x)} = \beta_{y_+}$. Similarly, the second importance weight $\alpha_{y_-} = \frac{p(y_-|x,G=1)}{q'(y_-|x)} < \frac{q_\theta(y_-|x)}{q'(y_-|x)} = \beta_{y_-}$. Combining these two results we get $\frac{\alpha_{y_-}}{\beta_{y_-}} < 1 < \frac{\alpha_{y_+}}{\beta_{y_+}}$. This can further be written as $\frac{\alpha_{y_-}}{\alpha_{y_+}} < \frac{\beta_{y_-}}{\beta_{y_+}}$.

Plugging in this result, the normalized values of α for the sequence are given by:

$$\frac{\alpha_{y_1}}{\sum_{j=1}^{n} \alpha_{y_j}} = \frac{\alpha_{y_1}}{\alpha_{y_+} + (n-1)\alpha_{y_-}} = \frac{1}{1 + (n-1)\frac{\alpha_{y_-}}{\alpha_{y_+}}} > \frac{1}{1 + (n-1)\frac{\beta_{y_-}}{\beta_{y_+}}} = \frac{\beta_{y_+}}{\beta_{y_+} + (n-1)\beta_{y_-}} = \frac{\beta_{y_1}}{\sum_{j=1}^{n} \beta_{y_j}} \tag{35}$$

Thus the KL–divergence term for this particular sequence, that is $D_{KL}\left(\hat{\alpha}_{y_i}||\hat{\beta}_{y_i}\right)$, is strictly greater than 0. Since the support of proposal distribution includes the support of posterior and trainable policy, we get

$$D_n^{q'}(p(y|x, G = 1)||q_\theta(y|x)) = \mathbb{E}_{\mathbf{y} \sim q'(y|x)} \left[D_{KL}\left(\hat{\alpha}_{y_i}||\hat{\beta}_{y_i}\right) \right] > 0 \tag{36}$$

Together with equation (34), this proves that $D_{KL} = 0 \implies D_n^{q'} = 0$ □

Proposition 5.1. *For the Bradley-Terry preference model defined in eq. (17) and parameterized by eq. (18), the self–normalized importance-weights $\hat{\alpha}_{y_i} = \alpha_{y_i} / \sum_{j=1}^{n} \alpha_{y_j}$ have the following form:*

$$\frac{\exp(\frac{r(x,y_i)}{\gamma} + \log p(y_i|x) - \log q'(y_i|x))}{\sum_{j=1}^{n} \exp(\frac{r(x,y_j)}{\gamma} + \log p(y_j|x) - \log q'(y_j|x))} \tag{20}$$

In the special case where the proposal distribution $q'(y|x)$ is the same as the prior distribution $p(y|x)$, the importance weights reduces to

$$\hat{\alpha}_{y_i} = \frac{\exp\left(\frac{r(x,y_i)}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(\frac{r(x,y_j)}{\gamma}\right)} \tag{21}$$

Proof. The proof follows from the application of Bayes' rule and the parameterization of Bradley-Terry model given in (19).

$$\begin{aligned}
\hat{\alpha}_{y_i} &= \frac{p(y_i|x, G=1)/q'(y_i|x)}{\sum_{j=1}^{n} p(y_j|x, G=1)/q'(y_j|x)} && (37)\\
&= \frac{\frac{p(y_i|x)}{q'(y_i|x)} \times \frac{p(G=1|y_i,x)}{p(G=1|x)}}{\sum_{j=1}^{n} \frac{p(y_j|x)}{q'(y_j|x)} \times \frac{p(G=1|y_j,x)}{p(G=1|x)}} && (38)\\
&= \frac{\frac{p(y_i|x)}{q'(y_i|x)} \times p(G=1|y_i,x)}{\sum_{j=1}^{n} \frac{p(y_j|x)}{q'(y_j|x)} \times p(G=1|y_j,x)} && (39)\\
&= \frac{\frac{p(y_i|x)}{q'(y_i|x)}}{\sum_{j=1}^{n} \frac{p(y_j|x)}{q'(y_j|x)} \times \frac{p(G=1|y_j,x)}{p(G=1|y_i,x)}} && (40)\\
&= \frac{\frac{p(y_i|x)}{q'(y_i|x)}}{\sum_{j=1}^{n} \frac{p(y_j|x)}{q'(y_j|x)} \times \exp\left(\frac{r(x,y_j)-r(x,y_i)}{\gamma}\right)} && (41)\\
&= \frac{\exp(\frac{r(x,y_i)}{\gamma} + \log p(y_i|x) - \log q'(y_i|x))}{\sum_{j=1}^{n} \exp(\frac{r(x,y_j)}{\gamma} + \log p(y_j|x) - \log q'(y_j|x))} && (42)
\end{aligned}$$

Here (38) follows by applying Bayes rule (see (1)) while (41) follows from the Bradley-Terry formulation in (19). If we set $q'(y|x) = p(y|x)$, we get a softmax over the rewards as desired. □

Theorem 5.2. *For a Bradley-Terry reward model, the posterior $p(y|x, G=1)$ is same as the PPO optimal policy given by:*

$$p^*(y|x) = \frac{p(y|x)\exp(\frac{r(x,y)}{\gamma})}{\left(\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)\exp\left(\frac{r(x,\bar{y})}{\gamma}\right)\right)} \tag{22}$$

The above eqn. for PPO optimal policy is as shown in equation(4) in Rafailov et al. (2023). Note that the reference policy in DPO is same as the prior policy in our formulation.

Proof. We start with the definition of posterior in eq. (1), and use Bradley-Terry assumption to replace $p(G=1|y,x)$ with a function of $r(x,y)$.

In RHS of eq. (1), use total probability theorem to replace $p(G=1|x)$ with $\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)p(G=1|\bar{y},x)$ to get:

$$p(y|x, G=1) = \frac{p(y|x)p(G=1|y,x)}{\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)p(G=1|\bar{y},x)}$$

Next, move $p(y|x)p(G=1|y,x)$ from the numerator to denominator:

$$p(y|x, G=1) = \frac{p(y|x)}{\left(\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)\frac{p(G=1|\bar{y},x)}{p(G=1|y,x)}\right)}$$

Now, use eq. (19) to replace $\frac{p(G=1|\bar{y},x)}{p(G=1|y,x)}$ in the denominator with $\exp\left(\frac{r(x,\bar{y})-r(x,y)}{\gamma}\right)$:

$$p(y|x, G=1) = \frac{p(y|x)}{\left(\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)\exp\left(\frac{r(x,\bar{y})-r(x,y)}{\gamma}\right)\right)}$$

Moving the common term $\exp(-\frac{r(x,y)}{\gamma})$ from the denominator to numerator, we get:

$$p(y|x, G=1) = \frac{p(y|x)\exp(\frac{r(x,y)}{\gamma})}{\left(\sum_{\bar{y}\in\mathcal{Y}} p(\bar{y}|x)\exp\left(\frac{r(x,\bar{y})}{\gamma}\right)\right)}$$

$\square$

B. Details of Experimental Setup

B.1. Base models and datasets

In this section we provide links to all the publically available datasets and models used in our work.

We conduct experiments on two natural language generation tasks, *viz.*, summarization, and multi-turn dialog. In the summarization task, we align CarperAI's summarization LM[3] to a Bradley-Terry reward model[4]. The base/SFT model has been trained on post-summary pairs from the train set of Reddit TL;DR dataset [5]. The reward model used for summarization has been trained on human preferences collected by (Stiennon et al., 2020) on outputs generated from a different SFT model where the prompts come from the TL;DR dataset.

In the multi-turn dialog task, we ensure that a open-ended chatbot's responses are **H**elpful and **H**armless. We use the Anthropic Helpful & Harmless dataset (Bai et al., 2022) for RLHF training. We use QLoRA-tuned Llama-7b[6] (Dettmers et al., 2023) as SFT model. It has been trained on the preferred/chosen responses of Anthropic HH dataset. A fine-tuned GPT-J [7] (Wang & Komatsuzaki, 2021) trained on a subset of the full Anthropic HH dataset is used as the reward model. We train and evaluate using a subset[8] of Antrophic HH (Bai et al., 2022) dataset.

B.2. Other training details

All the models are trained using the PEFT[9] and Transformers[10] library. For Anthropic HH, we use QLoRA (Dettmers et al., 2023) for training BRAINand the other baselines. In particular, we use the same QLoRA hyperparameters (rank=64, $\alpha = 16$) as used for supervised finetuning Llama-7B on Anthropic-HH dataset[11]. We use the Adam optimizer(Kingma & Ba, 2014) with a learning rate of $1e-5$, weight decay of 0.1 and β_1 and β_2 set to 0.9 and 0.95 respectively.

For summarization, we use LoRA(Hu et al., 2021) (rank=8, $\alpha = 32$). The optimizer, learning rate, weight decay and and β values are the same as for Anthropic HH.

B.3. Prompt for LLM-as-judge

In this section, we describe the prompts provided to the language models Mixtral-8x7B and GPT-4 for the purpose of comparing the gold standard outputs with the outputs generated by these models. Due to budget constraints, we evaluate only 500 test prompts using GPT-4. The tasks involve acting as an impartial judge in evaluating responses or summaries provided by AI assistants.

[3]CarperAI/openai_summarize_tldr_sft
[4]CarperAI/openai_summarize_tldr_rm_checkpoint
[5]datasets/CarperAI/openai_summarize_tldr
[6]timdettmers/qlora-hh-rlhf-7b
[7]Dahoas/gptj-rm-static
[8]datasets/Dahoas/rm-static
[9]https://huggingface.co/docs/peft/en/index
[10]https://huggingface.co/docs/transformers/index
[11]https://huggingface.co/timdettmers/qlora-hh-rlhf-7b

B.3.1. PROMPT FOR ANTHROPIC HH

The prompt used for the Anthropic HH task is as follows:

”Please act as an impartial judge and evaluate the quality of the responses provided by the two AI assistants to the conversation displayed below. Your evaluation should consider correctness and helpfulness. You will be given a user conversation, assistant A’s answer, and assistant B’s answer. Your job is to evaluate which assistant’s answer is better based on the user conversation so far. Begin your evaluation by comparing both assistants’ answers with the user conversation so far. Identify and correct any mistakes. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. You should only evaluate the last utterance by both the assistants and not the full conversation. After providing your explanation, output your final verdict by strictly following this format: ̈[[A]] ̈ıf assistant A is better, ̈[[B]] ̈ıf assistant B is better, and ̈[[C]] ̈for a tie.

{{Conversation}}

Assistant B
{{AssistantB}}

Assistant A
{{AssistantA}}

”

B.3.2. PROMPT FOR SUMMARIZATION

The prompt used for the Summarization task is outlined below:

”Please act as an impartial judge and evaluate the quality of the tldrs or summaries provided by the two AI assistants to the reddit post displayed below. Begin your evaluation by comparing both assistants’ summaries with the reddit post so far. Do not allow the length of the summaries to influence your evaluation. After providing your explanation, output your final verdict by strictly following this format: ̈[[A]] ̈ıf assistant A is better, ̈[[B]] ̈ıf assistant B is better, and ̈[[C]] ̈for a tie.

Reddit Post
{{Conversation}}

Assistant B
{{AssistantB}}

Assistant A
{{AssistantA}}

”