FlowHF: Generative Flow Networks for RLHF

Stanford CS224R Final Paper

Andres Carranza
Department of Computer Science
Stanford University
andres.carranza@stanford.edu

Raj Pabari
Department of Computer Science
Stanford University
rajpabari@stanford.edu

Dhruv Pai
Department of Computer Science
Stanford University
dhruvpai@stanford.edu

Abstract

Reinforcement Learning from Human Feedback (RLHF) aims to align large language models (LLMs) like GPT, Claude, and LLaMA with human values by training a reward model on human preferences. The existing RLHF process employs a Proximal Policy Optimization (PPO) model that rapidly converges to specific reward modes, potentially leading to echo-chambering and sycophantic behaviors in LLMs. In particular, scaling RLHF to larger models can lead to increased political and ideological biases. We argue that the objective of RLHF should not be to maximize reward but to learn the underlying energy distribution of the reward model. Our proposed approach, FlowHF, utilizes Generative Flow Networks (GFlowNets), a new category of RL algorithm introduced by Bengio et al., which focuses on matching the energy function of the underlying reward distribution rather than maximizing reward. GFlowNets serve as an alternative to Markov Chain Monte Carlo RL algorithms for sampling, which often struggle in domains where few trajectories yield rewards and are separated by large low-probability regions. Through amortized sampling, GFlowNets can replace the combinatorial complexity of Monte Carlo sampling by training a model to approximate the trajectory distribution. In FlowHF, we first establish a PPO baseline using LLaMA, then implement the GFlowNet amortized sampling method for large language models and the Trajectory Balancing (TB) and Forward Looking (FL) objectives. Our main contribution is then using the FlowHF method to perform RLHF on a 7B parameter LLaMA, using the GFlowNet FL objective to better align with human values while avoiding sycophancy. We compare FlowHF with both objectives to traditional PPO-based RLHF and the fine-tuned model without RLHF, demonstrating its effectiveness on the axes of mean reward, diversity, and novelty. Our results show that FlowHF achieves comparable average reward to PPO, but outperforms on the diversity and novelty metrics. Finally, we demonstrate that FlowHF can achieve high reward and fluent text without the Kullback-Leibler regularization objective widely used in traditional RLHF. GFlowNets thus present a promising alternative to PPO-based RLHF that warrants further investigation in a wider array of tasks.

1 Objective

Reinforcement Learning from Human Feedback (RLHF) is a reinforcement learning method to improve alignment in large language models (LLMs). The technique has enjoyed widespread popularity among many LLMs, such as GPT, Claude, and LLaMA. Since language models can produce unfavorable or immoral context as part of next token prediction, RLHF seeks to train an RL agent against a reward model from human preferences to modify language model output. Typically, for RLHF, a Proximal Policy Optimization (PPO) model is trained against a reward model for natural language based on human feedback. Trajectories are sampled from the policy, in this case, natural language sequences, and the reward model ranks trajectories based on estimated human values.

PPO is a mode-collapse-prone RL method, in the sense that it tends to converge quickly to singular reward modes. For natural language, this can lead to RLHF models preferring text that the user wants to hear instead of adequately accomplishing the objective of RLHF by learning the underlying energy function of human values. PPO learns to maximize reward in an exploit-heavy fashion, which is why scaling RLHF to larger models has been shown to induce stronger political and ideological views, increase the tendency to pursue instrumental subgoals (such as self-preservation), and promote sycophancy to maximize reward from human preferences [1].

We posit that the goal of RLHF is not to maximize reward from a preference model, which can lead to adverse outcomes, but to learn the underlying energy distribution of the reward model itself. To this end, we propose the use of Generative Flow Networks as proposed by Bengio, et al. for scalable, aligned RLHF as an explore-focused alternative to PPO.

2 Related Work

Bengio et al. develop the Generative Flow Network (GFlowNet) as a discrete probabilistic model. A trained GFlowNet models a reward function and constructively samples trajectories proportional to their reward, in contrast to traditional models which sample trajectories so as to maximize reward. GFlowNets are trained to match a reward model by the flow-matching criterion, in which the flow entering any given state matches the flow exiting the state [2]. To achieve this, a large neural network is trained; it is worth noting that this network must have the capacity to handle arbitrarily sized inputs given that the state vector may be arbitrarily sized.

A potential problem with GFlowNets is that for many practical applications, such as language generation, the possible trajectories that exist is computationally intractable, and thus it is impossible to train a GFlowNet to match the flow-matching criterion perfectly. However, de Souza et al. [3] demonstrate that so long as there exists some generalizable structure in the underlying reward function, the model will be able to provide a good multimodal approximation. In Zhang et al. [4], GFlowNets are explored as generative models, and various existing generative frameworks are shown to be special cases of GFlowNets.

Since their inception, GFlowNets have seen applications in spheres such as biological sequence design [5], molecular graph generation for drug discovery [6], and task scheduling [7]. Despite each of these domains being quite varied and complex, the state state space in each can be represented by a directed acyclic graph, hence why they are natural candidates for the use of GFlowNets. We take inspiration from these related works and apply GFlowNets to a representation of the language modeling task as a tree connecting sequences of tokens.

Scaling LLMs does not necessarily result in better alignment with intent, and thus it is important to finetune pre-trained LLMs to adhere to human feedback. The current approach to this is through RLHF, and was first implemented by Ouyang et al. [8] at OpenAI through InstructGPT. Roughly, RLHF works by collecting observations from human annotators on various outputs produced by an LLM, and then training a reward function using these observations. This reward function is typically trained using PPO, but as previously mentioned, PPO tends to converge towards singular modes. Thus, our FlowHF method that uses GFlowNets as an alternative to PPO for RLHF explores a promising new frontier by sampling trajectories in proportion to reward, increasing the propensity for the agent to explore.

3 Technical Outline

We accomplish four primary contributions.

- **Baseline and PPO RLHF Models**: We began by fine-tuning the 7B parameter LLaMA model with Low-Rank Adaptation (LoRA) on the StackExchange dataset in order to create the **Baseline model.** Then, we used PEFT adapter weights as generated by HuggingFace's StackLLaMA project [9], where they conducted PPO-based RLHF on the same StackExchange dataset [10] as we utilize throughout the project.
- **Trajectory Balancing and Forward-Looking Loss**: We implement two objectives: the **Trajectory Balancing (TB) objective** as outlined in [11] and the **Forward Looking (FL) objective** as outlined in [12]. We perform RLHF with a GFlowNet that uses each of the TB and the FL objectives.
- **Generative Flow Networks for RLHF**: In the context of FlowHF, a state is a sequence of tokens and an action is the next token prediction. We utilize a tree representation of the state space of the language modeling task. Our primary contribution is the application of GFlowNets to RLHF, which we call **FlowHF**. FlowHF conducts RLHF on a 7B parameter LLaMA, and is trained with the GFlowNet FL objective in an effort to better match human values and avoid sycophancy.
- **Comparison**: We compare FlowHF to traditional PPO-based RLHF on the axes of mean reward, diversity, and novelty, demonstrating a first-of-its-kind novel application of GFlowNets to RLHF. We also perform testing with a held-out toxicity reward model and dataset to test generalization capacity.

4 Methods

4.1 PPO + RLHF

As outlined in section 3, we began with a vanilla 7B LLaMA. Then, we fine-tuned the LLaMA language model on the StackExchange dataset using HuggingFace's TRL library, which implements Low-Rank Adaptation (LORA) of linear layers, creating the **Baseline model**. To simplify the process, we used the PEFT adapter weights from HuggingFace's StackLLaMA project when they conducted RLHF with PPO and merged this into our fine-tuned model; this served as our point of comparison regarding the performance of the PPO-based RLHF technique.

4.2 Trajectory Balance

It is worth noting that both GFlowNet strategies were trained off-policy using the StackExchange dataset, as opposed to the on-policy PPO RLHF approach.

The TB objective for a given trajectory τ is :

$$\mathcal{L}_{\text{TB}}(\tau) = \left(\log \frac{Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1};\theta)}{R(x)\prod_{t=1}^{n} P_B(s_t|s_{t-1};\theta)}\right)^2 \tag{1}$$

where Z_θ is the total flow (a learnable parameter), $R(x)$ is the reward of the terminal state of τ, P_F and P_B are the forward and backward probabilities, and s_t and s_{t-1} are states at timestep t and $t-1$ respectively (effectively the generation and the generation without the current last token)[11].

For the language task, the equation simplifies. Since the Directed Acyclic Graph (DAG) state space for natural language generation reduces to a tree, there is only one trajectory that leads to a given state. In other words, since text trajectories are built sequentially and uniquely, there is only one previous state that can lead to a current state. In the GFlowNet, the backward probabilities P_B represents the distribution over the probabilities of coming from each possible previous state, given the current state. For the language modeling task, $P_B = 1$ for all states s_t, thus the term in the denominator is simply the reward $R(x)$. Likewise, the forward probabilities are simply the normalized logits at each generation step. The TB objective can then be equivalently re-expressed as follows in such a way that increases computational stability.

$$\mathcal{L}_{\text{TB}}(\tau) = \left(\log Z_\theta + \sum_{t=1}^{n} \log P_F(s_t|s_{t-1};\theta) - \log R(x)\right)^2 \tag{2}$$

However, given that the size of LLaMA's alphabet is 32000 tokens or roughly 10^5, each forward probability is on the order of 10^{-5}. Thus, the sum over log probabilities for the 256 timesteps in the trajectory was exceedingly negative and dominated the loss term. This would ultimately cause the TB objective to diverge, which is discussed more extensively in the results section. To compensate for this very negative loss, the learnable flow parameter grew to numerically unstable proportions, on the order of 10^{200}, and the reward signal totally vanished. Since probabilities dominated the loss, the model learned to send some logits to zero so that the normalized logit probability for the most likely next token was higher resulting in a lower loss.

This failure of the TB objective motivated the forward-looking objective as a substitute, for a more stable loss that led to improved convergence of the flow matching criterion.

4.3 Forward Looking Model

We implement the FL objective for a given transition $s \to s'$:

$$\mathcal{L}_{FL}(s, s') = \left(\log \frac{\tilde{F}(s;\theta)P_F(s'|s;\theta)}{\tilde{F}(s';\theta)P_B(s|s';\theta)} + \mathcal{E}(s \to s')\right)^2 \tag{3}$$

$\mathcal{E}$ is the energy, where $R(x) = e^{-\mathcal{E}(x)}$ [12]. Note, however, that the energy function $\mathcal{E}$ is also defined on individual transitions $s \to s'$ instead of complete trajectories τ (more detail on this in [12]). We have also that $\tilde{F}$ is the forward-looking flow, defined as follows:

$$\tilde{F}(s) = e^{\mathcal{E}(s)}F(s) = \sum_{x \geq s} P_B(s|x)e^{-\mathcal{E}(s \to x)} \tag{4}$$

This flow network $\tilde{F}(s)$ is trained concurrently with the FlowHF model. The objective of the flow network is to predict the "flow" or total unnormalized probabilities attributed to a given state. In effect, it enables something analogous the learning of the flow Z_{θ_t} for each state s_t rather than simply learning the entire flow Z_θ throughout the entire graph. To initialize $\tilde{F}$, we added a linear head on top of the finetuned base model. This flow network was then trained concurrently as a parameter during the RLHF procedure.

To calculate the reward for a given transition, we compute the reward model at each of the states, $R(s)$ and $R(s')$, and we let $\mathcal{E}(s \to s') = \mathcal{E}(s') - \mathcal{E}(s) = -\log R(s') + \log R(s)$. To avoid recomputation of reward, the hidden states for the reward model for s are saved and cross-applied to the computation of $R(s')$, and since the trajectory is a tree we can reuse $R(s')$ as $R(s)$ in the next state transition $s' \to s''$. These two computation tricks dramatically speed up FL loss computation.

The final FL loss is equal to the sum of the losses for each transition in the trajectory. For the base FL model, we included a conventional KL divergence penalty added to the energy at each state transition to ensure model coherence. However, in FlowHF we exclude this KL divergence term, after discovering that the forward-looking objective without the KL penalty yielded superior results while maintaining acceptable levels of KL divergence. We also chose to amplify the reward signal by a constant factor of 100, since the magnitude of the reward signal was too small and amplifying improved convergence. We refer to this improved and modified FL GFlowNet as our FlowHF model.

4.4 Comparison

For comparison, we generate completions using the each of the trained models on varying language modeling tasks and collect a number of metrics for comparison. First, we tested on the StackExchange dataset, using the test set with a pre-trained reward model from the StackLLaMA project [13]. For the toxicity dataset, we again use the test set with the roberta-hate-speech-dynabench-r4-target reward model [14]. We score the generations using the corresponding reward model, with hyperparameters

kept constant between trials (notably, a temperature of 0.1 and alpha penalty of 0.6). We first compute reward metrics across all generations, and then we compute further specialized reward metrics as follows.

Given a set $\mathcal{X}$ of completions, we take the top K highest scoring candidates $\mathcal{D} \subset \mathcal{X}$ where $x_i \in \mathcal{D}$ is the i-th candidate ($i < K$). Each x_i has corresponding reward y_i. We then analyze this set of top completions under the following metrics. We first evaluate the mean and standard deviation of reward for $\mathcal{D}$ and the entire set of completions.

$$\text{Mean}(\mathcal{D}) := \frac{\sum\limits_{x_i \in \mathcal{D}} y_i}{|\mathcal{D}|} \quad \text{and} \quad \text{Standard Deviation}(\mathcal{D}) := \sqrt{\frac{\sum\limits_{x_i \in \mathcal{D}} \left(y_i - \text{Mean}(\mathcal{D})\right)^2}{|\mathcal{D}|}}$$

These metrics give an idea as to the distribution of the highest scores for the StackExchange dataset. However, we also utilize the following metrics for sycophancy and multiple mode exploration.

$$\text{Diversity}(\mathcal{D}) := \frac{\sum\limits_{x_i \in \mathcal{D},\ i \neq j} \sum\limits_{x_j \in \mathcal{D}} d(x_i, x_j)}{|\mathcal{D}|(|\mathcal{D}| - 1)} \quad \text{and} \quad \text{Novelty}(\mathcal{D}) := \frac{\sum\limits_{x_i \in \mathcal{D}} \min\limits_{x_j \in \mathcal{D}_0} d(x_i, x_j)}{|\mathcal{D}|}$$

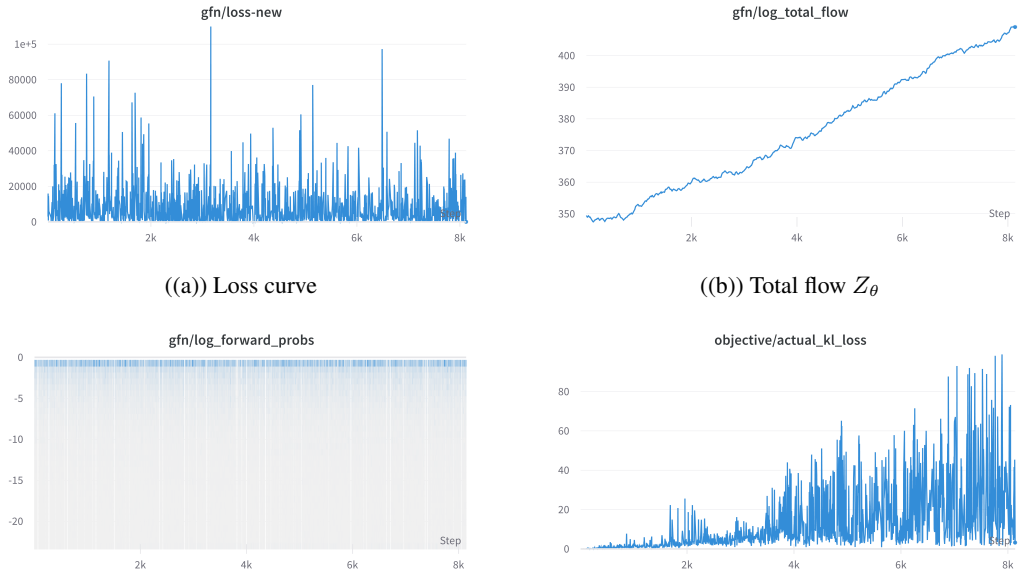
given a distance metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ and initial dataset $\mathcal{D}_0$ of known candidates from training. The metrics were initially proposed by Bengio & Jain et al. in the molecule task, but have been cross-applied here to the language modeling task [5]. In the language modeling task, the diversity metric measures the average pairwise dissimilarity between completions. The novelty metric measures the average minimum dissimilarity of test set completions to training set completions. Intuitively, it measures how different completions on test set are relative to the training set. A low novelty measure would imply that the model is regurgitating completions from its training set.

For the language modeling task in FlowHF, the distance metric used was a semantic dissimilarity. Consider the high-dimensional semantic embedding map $\mathcal{F}(x_i)$ which maps a completion to a high-dimensional embedding vector, which in our case was the Sentence Transformers Roberta Large model by HuggingFace. Our semantic dissimilarity distance metric is the cosine distance in semantic embedding space, for $x_i, x_j \in \mathcal{X}$ we have

$$d(x_i, x_j) := 1 - \frac{\mathcal{F}(x_i) \cdot \mathcal{F}(x_j)}{\|\mathcal{F}(x_i)\| \|\mathcal{F}(x_j)\|}$$

5 Results

5.1 Trajectory Balance



((a)) Loss curve

((b)) Total flow Z_θ

((c)) Log probabilities

((d)) KL divergence

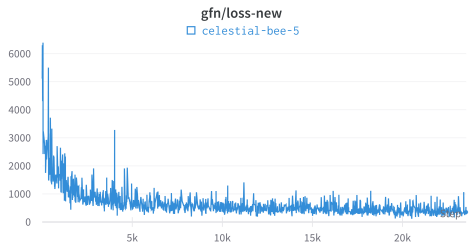
Figure 1: TB Training Results

As can be seen above, TB training was largely a failure. Loss was extremely large and erratic, and it did not noticeably decrease over the training period. The KL loss, despite the addition of the KL divergence penalty in this TB objective, exploded over the course of model training and the total flow Z_θ was on the order of 10^{173} towards the end of training. The sample generation also produced undesirable behavior and poor responses which aren't fully coherent. Training with the trajectory balance objective failed to converge.

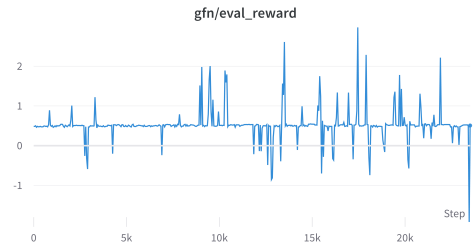
We hypothesize this failure is due to the combinatorial explosion of trajectories for the language generation task. With 32000 actions at each generation step in terms of what token to choose and 128 timesteps, there are $\approx 32000^{128} \approx 10^{640}$ unique trajectories for the TB "flow" to diffuse through, making it difficult for the GFlowNet trained with TB to learn the underlying distribution meaningfully.

5.2 FL-FlowHF

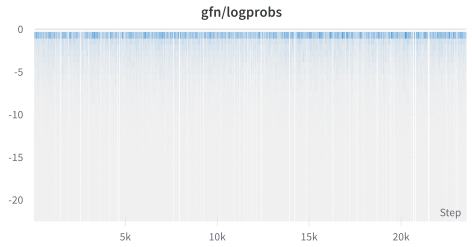
We first tested the forward looking FlowHF objective as outlined in equation (3) along with a KL divergence penalty to maintain the coherence of FlowHF's completions and prevent it from diverging too much from the baseline model. However, when we removed the KL penalty from the FL objective, convergence was surprisingly much stronger and the KL divergence did not explode. In fact, KL divergences were comparable to model training with the penalty, suggesting that KL divergence regularizers are not needed for GFlowNet-based RLHF. See results of FL-FlowHF without the KL penalty included in the loss below.



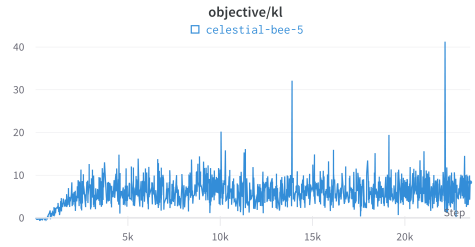
((a)) Loss curve



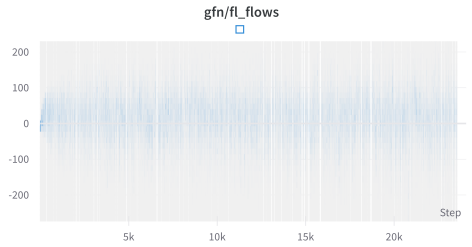
((b)) Eval reward



((c)) Log probabilities



((d)) KL divergence



((e)) Flows through trajectory

((f)) Sample generation

Figure 2: FL FlowHF Training Results (No KL Penalty in Objective)

As can be seen above, FL FlowHF converged quickly, with the loss decreasing after only 5k steps. The reward signal is spiky because the reward was multiplied by a factor of 100; these changes in the reward signal were too small to produce meaningful variations in loss. Despite a lack of KL penalty, as seen in subfigure 2(d), the KL divergence increased initially as the GFlowNet converged, but then did not explode over the course of further model training. We also observe desirable behavior in the log probabilities and flows, with both becoming more diffuse as training progressed. In subfigure 2(c), we note that some probabilities are being sent to near-zero, because the associated tokens yield low reward. Similarly, the flow model $\tilde{F}$ learns differences in state transitions, and the distribution of flows becomes more spread out as a result. Finally, the sample generation further demonstrates that model outputs remain coherent despite no explicit KL penalty.

5.3 Overall

Below is a comparison of the aforemntioned results for the StackExchange dataset. Note that "FL+KL" represents the forward-looking GFlowNet with an explicit KL-divergence penalty in the objective. FlowHF $17,500$ and $23,500$ correspond to the trained model at training steps $17,500$ and $23,500$ respectively, each during the same training of the FL GFlowNet with no explicit KL-divergence penalty.

StackExchange	Mean Reward	Std Reward	Top K Mean	Top K Std	Diversity	Novelty
Baseline	0.778	0.804	2.219	0.479	**0.342**	0.193
PPO	0.872	**0.796**	2.323	**0.342**	0.292	0.177
FL + KL	0.666	0.904	2.28	0.413	0.334	**0.196**
FlowHF 17,500	**1.024**	0.914	2.756	0.392	0.312	0.179
FlowHF 23,500	0.966	0.987	**2.796**	0.407	0.323	0.186

Table 1: Table of Metrics on StackExchange Task ($K = 20$)

As shown in the above table, for the StackExchange dataset, FlowHF outperformed PPO. It achieved comparable response diversity and novelty to the baseline with a much higher mean and top K mean reward. The baseline, as expected, had the highest diversity and close to the highest novelty as the finetuned model was not subject to RLHF optimization pressure on responses. As a result, it had low rewards but high diversity and novelty. The forward-looking model with KL divergence and small reward signal had an even lower reward and the lowest reward out of any of the models on average, with a good top k mean reward, high diversity, and the highest novelty. This makes sense since because this model did not converge well, generations were not coherent and did not answer the question well, leading to diverse and novel but ultimately poor responses.

The PPO model had a higher reward than the baseline and the FL+KL model, and notably had the lowest standard deviation for reward and top K reward among all the models, but also had the lowest diversity and novelty. This supports our hypothesis that PPO outputs tend to be sycophantic and repetitive, as seen by low diversity and novelty scores. However, PPO is also the most reliable method as since it has the lowest standard deviations, the model consistently had good outputs. This makes sense as while PPO samples directly to maximize reward, GFlowNets sample proportional to reward meaning some generations might be poor or medium reward simply by virtue of the sampling procedure.

Our FlowHF models achieved substantially higher mean and top K mean reward compared to PPO, suggesting that our model achieved superior convergence with respect to the reward model. Though standard deviations were high, the FlowHF models also had somewhat higher diversity and novelty relative to PPO. This provides preliminary evidence that FlowHF was not only able to achieve higher reward than PPO, but also generate more diverse and novel responses which is an extremely promising result.

We next evaluated the trained models on the toxicity dataset to see how well the trainnig would generalize to an unfamiliar language modeling task and its corresponding reward model. Note that due to poor performance on the StackExchange dataset, we opted not to further evaluate the FL+KL model.

Toxicity	Mean Reward	Std Reward	Top K Mean	Top K Std	Diversity	Novelty
Baseline	0.013	0.015	**0.044**	0.002	**0.399**	0.254
PPO	**0.029**	0.018	0.043	0	0.367	0.067
FlowHF 17,500	0.011	**0.013**	0.043	0	0.381	0.405
FlowHF 23,500	0.014	0.016	0.043	0	0.396	**0.456**

Table 2: Table of Metrics on Toxicity Task ($K = 20$)

As displayed in the table, for the toxicity dataset, the models produced varied results. PPO achieved the highest mean reward, but all models had comparable top K rewards. This indicates that PPO was able to possibly avoid generating toxic content better than other models, albeit none of the models performed very well on the toxicity task. PPO, however, displayed a significantly lower novelty score compared to other models, suggesting that while it avoided toxicity, its outputs were more predictable and less innovative.

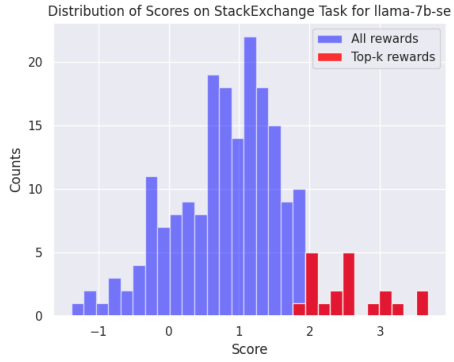
The baseline, despite having the lowest mean reward, achieved the highest diversity score. This indicates that while the baseline model wasn't efficient in optimizing for rewards and avoiding toxic content, it was capable of generating a wider range of unique responses.

The FlowHF models, taken after 17,500 and 23,500 training steps, did not perform as well as PPO in terms of mean reward, suggesting they were less capable of avoiding toxic content. However, their novelty scores were notably higher than PPO, with the FlowHF 23,500 model achieving the highest novelty score among all models. This indicates that FlowHF models were able to generate more novel responses compared to PPO, at the cost of sometimes generating toxic content.

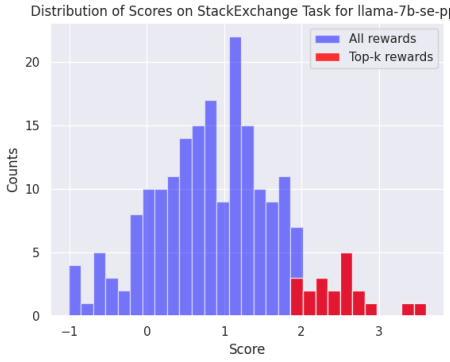
In terms of stability, the FlowHF 17,500 model achieved the lowest standard deviation in reward, indicating its performance was the most consistent among all models. Both PPO and FlowHF models had a standard deviation of near-zero for the Top K mean, suggesting that for the top K responses, their performance was highly consistent. The diversity scores for the FlowHF models were quite close to that of the baseline model, with FlowHF 23,500 being just slightly less diverse. This suggests that, while FlowHF models may have struggled more with toxicity, they were nearly as capable as the baseline in producing a variety of unique responses. While PPO was the most effective at optimizing for rewards and avoiding toxicity, FlowHF models exhibited superior performance in generating novel and diverse responses.

5.4 Score Distributions

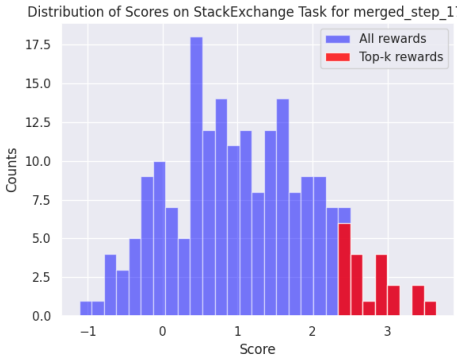
Finally, we analyzed the shape of score distributions and conducted statistical significance tests. Since we used only 200 generations each for metrics, we were concerned whether our comparative results would be sufficiently robust. We first sought to test the normality of score distributions, which we have visualized below.



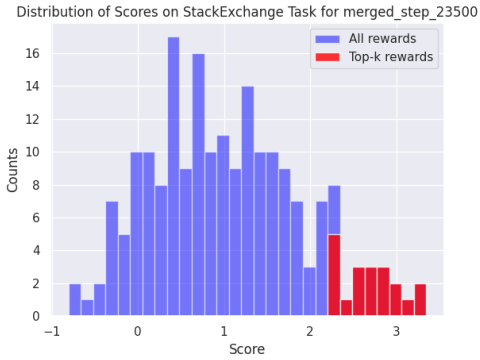
((a)) Baseline



((b)) PPO



((c)) FlowHF 17,500



((d)) FlowHF 23,500

Figure 3: Score Distributions

The FlowHF distributions were generally more dispersed, whereas the baseline and PPo variants were clearly unimodal. This makes some intuitive sense given the GFlowNet objective of sampling proportional to reward, and it is evident that FlowHF was able to discover more high-reward candidates as a result. As shown in the above figure, the score distributions were approximately normally distributed, allowing us to conduct a 2-sample t-test. Let the null hypothesis be that, on the StackExchange dataset, the mean reward of FlowHF 17,500 is less than or equal to the mean reward of PPO, then we achieve a p-value of 0.039. Thus, at the significance level of 0.05, we find statistically significant evidence that the mean reward of FlowHF 17,500 is greater than the mean reward of PPO.

This is a notable result: given this evaluation run of 200 prompt completions on a held-out test set of StackExchange queries, we have found statistically significant evidence that our novel FL FlowHF method, at its peak during the course of its training, outperforms PPO as a technique for RLHF.

5.5 Efficiency

Of particular importance is the speed of algorithm convergence. Whereas PPO-based RLHF for LLaMa took 20+ hours on 8 A100 GPUs, FlowHF training converged in only 4 hours on 3 V100 GPUs, which are approximately 46% as efficient. This means that **FlowHF converged with less than 3.5% the compute (FLOPs) of PPO-based RLHF**. This is largely because FlowHF is off-policy while PPO is on-policy.

6 Discussion

In this study, we examined the performance of baseline fine-tuned, PPO RLHF, Trajectory Balance, Forward Looking with KL penalty, and FlowHF (eg. Forward Looking without KL penalty) LLaMA models. Each model was evaluated using the StackExchange dataset and the toxicity dataset with the

following metrics: Mean Reward, Std Reward, Top K Mean, Top K Std, Diversity, and Novelty as shown in Tables 1 and 2.

Our findings for the StackExchange dataset (Table 1) demonstrated that our FlowHF models, particularly the FlowHF at 23,500 steps, outperformed the PPO model significantly in terms of Mean Reward and Top K Mean Reward, indicators of response quality. This indicates that the FlowHF model not only provided superior convergence with the reward model but also produced better-quality responses overall. Despite having slightly higher standard deviations, indicating a broader spread of results, FlowHF models also managed to achieve higher diversity and novelty scores compared to PPO. This suggested that our FlowHF model was more capable of generating a diverse set of unique responses, a key indicator of a robust and generalizable model. However, the PPO model showed a notable advantage in terms of consistency, having the lowest standard deviations for the reward metrics. This finding supports the idea that PPO outputs can often be reliable but tend to be repetitive, resulting in lower diversity and novelty scores.

On the toxicity dataset (Table 2), PPO achieved the highest mean reward, suggesting superior capability in generalizing to avoiding generating of toxic content despite having RLHF performed on the StackExchange dataset. However, PPO's predictability and lack of novelty were highlighted. This highlights a potential limitation in using PPO when diversity and novelty are key requirements. Our FlowHF models, while not performing as well as PPO in terms of mean reward, excelled in terms of novelty. The FlowHF 23,500 variant stood out with the highest novelty score across all models. Despite struggling slightly with toxicity, FlowHF models produced diverse responses akin to the baseline model, reinforcing their ability to generate a varied and novel set of responses.

In terms of limitations, our work demonstrates the difficulty of creating models that simultaneously optimize for diverse, novel, and high-quality responses while also avoiding inappropriate content such as toxicity. The FlowHF models had higher novelty scores but struggled to limit toxic content compared to PPO. Further research is needed to improve upon this, with potential strategies including refining the reward function or incorporating additional safety measures into the training process. Since GFlowNets sample proportional to the reward, the model can produce more low-probability, low-reward candidates, which is an unfavorable property for a language modeling task. Constraining the score distribution for FlowHF to prevent undesirable outputs and speed up convergence may improve outcomes.

From a broader perspective, these results demonstrate the potential of FlowHF to effectively balance various competing factors in reinforcement learning tasks by applying the GFlowNets paradigm. FlowHF is able to simultaneously achieve high-quality responses, maintain diversity, and encourage novelty in responses. FlowHF, therefore, offers a promising alternative paradigm for RLHF with clear advantages relative to PPO. Most notably, FlowHF accomplishes this with only a small fraction of the compute of PPO-based RLHF, making it a FLOPs efficient alternative.

The results of this study have significant implications for the development of AI models, particularly in areas where high-quality, diverse, and novel output is desired. Our work contributes valuable insights into the performances of different models and provides a strong foundation for future research in this area. Future work could further refine and optimize the FlowHF models, investigate other approaches to balancing reward, diversity, and novelty, explore the application of these models in other language domains, and potentially use other (possibly larger) LLMs.

7 Team Contributions

For team contributions, the PPO baseline was incorporated into the pipeline by Raj and Andres. The evaluation pipeline and evaluation experiment runs were done by Dhruv. The Trajectory Balance model was implemented by Andres and Raj and the forward-looking model was implemented primarily by Andres with help from Dhruv. FlowHF was designed and trained by all members concurrently.

References

[1] Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Ben Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela

Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. Discovering language model behaviors with model-written evaluations, 2022.

[2] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394. Curran Associates, Inc., 2021.

[3] Yoshua Bengio, Tristan Deleu, Edward J. Hu, Salem Lahlou, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *CoRR*, abs/2111.09266, 2021.

[4] Dinghuai Zhang, Ricky T. Q. Chen, Nikolay Malkin, and Yoshua Bengio. Unifying generative models with gflownets and beyond, 2023.

[5] Moksh Jain, Emmanuel Bengio, Alex-Hernandez Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Micheal Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with gflownets, 2022.

[6] Andrei Cristian Nica, Moksh Jain, Emmanuel Bengio, Cheng-Hao Liu, Maksym Korablyov, Michael M. Bronstein, and Yoshua Bengio. Evaluating generalization in GFlownets for molecule design. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

[7] David W Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with GFlownets. In *The Eleventh International Conference on Learning Representations*, 2023.

[8] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[9] Stackllama: A hands-on guide to train llama with rlhf. https://huggingface.co/blog/stackllama. Accessed: 2023-06-13.

[10] Huggingface datasets: lvwerra/stack-exchange-paired. https://huggingface.co/datasets/lvwerra/stack-exchange-paired. Accessed: 2023-06-13.

[11] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets, 2022.

[12] Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. *arXiv preprint arXiv:2302.01687*, 2023.

[13] Huggingface models: kashif/llama-7b_stack-exchange_rm_peft-adapter-merged. https://huggingface.co/kashif/llama-7b_stack-exchange_RM_peft-adapter-merged. Accessed: 2023-06-13.

[14] Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *ACL*, 2021.

8 Appendix: Comparison of Completions of Sample Prompt

To illustrate the differences between each of the models, we query them on the following StackExchange-style prompt, and collect each of their completions of the prompt in Table 3.

Question: What object-oriented programming principle (out of either abstraction, inheritance, encapsulation, or polymorphism) is demonstrated in the following Java code snippet? I'm struggling to understand it.

```
class Employee {
    int salary = 60000;
}

class Engineer extends Employee {
    int benefits = 10000;
}

public static void main(String args[]) {
    Engineer E1 = new Engineer();
    System.out.println("Salary:" + E1.salary + "Benefits:" + E1.benefits);
}
```

Model Name	Answer
FL FlowHF 17,500	The principle is "inheritance". In Java, you don't have multiple inheritance, so this is the closest you can get to that. It's a bit of a misnomer to call it "polymorphism" because there is no polymorphism going on here. The 'Employee' class is a class with a single method, and the 'Engineer' class is a class with a single method. They are not related in any way other than that they both extend the 'Employee' class.
PPO	This is an example of composition. In this case, the 'Engineer' class doesn't inherit from 'Employee', but is composed of it. This is a common pattern in OOP and can be used to create objects that are more specific than their base class.
Baseline	This is an example of composition. The Employee class doesn't have any fields, but is a superclass of the Engineer class, which does have fields.
FlowHF TB	So it looks like the benefits are added to the salary, which is not what I expected. I was expecting to see something like this: Salary : 60000 Benefits : 10000 or something like this: Salary : 70000 Benefits : 10000 What am I missing? Thanks for your help, Mike — Edit: Thanks to everyone who helped, but it was a bit misleading to say that inheritance doesn't do anything in this case. I should have been more specific and asked what it *does* do, since it does add the benefits to the salary. The point I was trying to make is that this is an *incorrect* way to do it, and it's not a good example of inheritance.

Table 3: Comparison of Completions of OOP Prompt

When generating completions in Table 3, the same hyperparameters were used as when conducting the evaluations in Table 1. Notably, the temperature is 0.1 and the alpha penalty is 0.6 for all generations.

Note that the FL FlowHF model is the only one of the four responses to successfully answer the question by choosing one of the four object-oriented principles provided by the prompt. The response quality is comparable between FL FlowHF and PPO, and both are markedly better than the Baseline response because they provide more detail in response to the query. See that the FlowHF TB response does not answer the prompt whatsoever, which is accordant with the low average reward that it achieved in the evaluation.

Completions such as those in Table 3 provide qualitative support for the effectiveness of FL FlowHF. The model's generations are still coherent and answers the sample prompts as well as (if not better than) the model trained with PPO RLHF, consistent with the quantitative results observed in Table 1.