LEARNING TO GENERATE BETTER THAN YOUR LLM

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) has emerged as a powerful paradigm for fine-tuning Large Language Models (LLMs) for text generation. In particular, recent LLMs such as ChatGPT and GPT-4 can engage in fluent conversations with users after finetuning with RL. Inspired by *learning-to-search* algorithms and capitalizing on key properties of text generation, we seek to investigate RL algorithms beyond general purpose algorithms like Proximal Policy Optimization (PPO). In particular, we extend RL algorithms to allow them to interact with a dynamic black-box guide LLM and propose RL *with guided feedback* (RLGF), a suite of RL algorithms for LLM fine-tuning. We experiment on the IMDB positive sentiment, CommonGen, and TL;DR summarization tasks. We show that our RL algorithms achieve higher performance than supervised learning (SL) and RL baselines, demonstrating the benefit of interaction with the guide LLM. On both CommonGen and TL;DR, we not only outperform our SL baselines but also improve upon PPO across a variety of metrics beyond the one we optimized for.

1 INTRODUCTION

Large Language Models (LLMs) have become very capable in various real-world applications ranging from being able to answer open-ended questions on numerous topics (Zhang et al., 2022), write articles from short descriptions (Goyal et al., 2022), generate code (Github, 2023), follow robot commands (Huang et al., 2022), solve puzzles (Bubeck et al., 2023), and even showcased as assistive models for education (Khan Academy, 2023) and healthcare (Lee et al., 2023b).

However, using supervised learning (SL) to train LLMs presents a challenging metric mismatch (Wiseman & Rush, 2016) between the training and testing regimes. The metric mismatch arises from the training metric being the log-loss while the testing metrics are task-specific such as BLEU or user satisfaction rating. This discrepancy is magnified when fine-tuning LLMs on downstream tasks where the main goal is not just producing fluent text but also being proficient at solving the specific task.

Reinforcement Learning (RL) by definition address this metric mismatch by directly optimizing the metrics through reward feedback. Recently, OpenAI fine-tuned LLMs with RL from human feedback (RLHF) to better align LLMs to human intentions, leading to the great success of ChatGPT (OpenAI, 2023). Recently, GRUE benchmark (Ramamurthy et al., 2022) systematically studied RL versus SL when finetuning LLMs on downstream tasks with predefined rewards. GRUE's preliminary results demonstrate the benefit of RL when fine-tuning LLMs, leading to the release of popular codebases such as RL4LMs (Ramamurthy et al., 2022), TRLx (CarperAI, 2023) and AlpacaFarm (Dubois et al., 2023), that enables RL for language models. However, ChatGPT, RL4LMs, TRLX, and AlpacaFarm all use vanilla policy gradient methods known to be sample inefficient and sensitive to local minima due to the combinatorially large search space of natural language generation (Ramamurthy et al., 2022).

In this work, we focus on more efficient ways of fine-tuning LLMs on downstream tasks with predefined rewards. Our approach is motivated by prior work on Imitation Learning (IL) for structured prediction, which often leverages an existing guide policy (not necessarily an optimal policy) to reduce the search space for more efficient and optimal learning. Our key observation is that since modern LLMs exhibit impressive general language capabilities, they can serve as guide policies to improve the RL procedure. Our framework, which we call, *RL with guided feedback* (RLGF), integrates a guide policy into a policy gradient framework. The guide policy can provide reasonable but sub-optimal predictions for downstream tasks, which our framework can then leverage to learn a near-optimal strategy. We introduce novel algorithms for fine-tuning LLMs using our RLGF framework while capturing various existing IL for structured prediction and RL algorithms.

We evaluate on three tasks. The first is IMDB where the goal is to generate a positive and fluent review given an initial context. The second is CommonGen where the goal is to write a fluent text that uses a given set of words. Finally, we test on the TL;DR summarization task where the objective is to learn to generate summaries using human preference data. For all tasks, we find evidence of metric mismatch from SL-based fine-tuning approaches and show that RL-based methods which utilize reward signals outperforms on the task metric. We then demonstrate RLGF outperforming PPO on reward, fluency, as well as automated lexical metrics such as Rouge. Finally, we investigate how various baselines and RLGF algorithms balance the inherent trade-off between reward optimization and the KL constraint in the RLHF objective. We provide both theoretical justification and empirical evidence to show the benefit of using feedback in RL for fine-tuning LLMs on downstream tasks.

2 Related Work

Here we present the most relevant works at the intersection of IL, RL, and natural language generation. Please see Appendix A for a more thorough treatment of the literature.

IL for Structured Prediction: Algorithms such as Schedule Sampling (SS) (Bengio et al., 2015), methods using SS (Duckworth et al., 2019; Mihaylova & Martins, 2019; Goyal et al., 2017), SEARNN (Leblond et al., 2017), Bridging the Gap (Zhang et al., 2019b), Mixer (Ranzato et al., 2015) been inspired by IL for structured prediction algorithms DAGGER (Ross et al., 2011), DAD (Venkatraman et al., 2015), and SEARN (Daumé et al., 2009). Our work is inspired by AggreVaTeD (Sun et al., 2017) (Differentiable AggreVaTe Ross & Bagnell (2014)) where the algorithm makes use of differentiable policies and multi-step feedback rather than immediate one-step predictions to imitate. Similarly, we present a differentiable version of LOLS (Chang et al., 2015) as well as an improvement, D^2LOLS.

LLM Fine-tuning from Human Preferences: Recent advancements in fine-tuning of Large Language Models (LLMs) have shown incredible success in tasks through learning from human preferences. Being simpler to accumulate human preferences, Reinforcement Learning from Human Feedback (RLHF) (Stiennon et al., 2020) introduced a paradigm to utilize RL to improve downstream performance on translation (Kreutzer et al., 2018b), summarization (Stiennon et al., 2020), storytelling (Ziegler et al., 2019), and instruction following (OpenAI, 2023). Although effective, following works have shown RLHF to be challenging due to reward hacking, difficulties in scaling, and training instability (Zhao et al., 2023; Rafailov et al., 2023; Liu et al., 2023). To circumvent these difficulties, recent works have proposed methods to optimize for human preferences without RL (Zhao et al., 2023; Yuan et al., 2023; Rafailov et al., 2023; Liu et al., 2023). DPO, SLiC, RRHF, and RSO are methods that optimize for compatibility with a preference dataset under a preference reward model such as the Bradley Terry model (Bradley & Terry, 1952). In contrast, our work takes a different approach to improving RLHF by investigating improvements to PPO (Schulman et al., 2017), the base RL algorithm used.

LLM Distillation: With an ever growing arsenal of powerful, black-box LLMs, recent work has aimed to distill specific capabilities into a smaller model. Knowledge distillation (Buciluǎ et al., 2006; Hinton et al., 2015) in autoregressive models investigated matching sequence level log probabilities (Kim & Rush, 2016), model hidden states (Jiao et al., 2019), or attention scores (Wang et al., 2020). Recently, more sophisticated methods, inspired from the IL literature, are being proposed to better imitate the expert LLM's performance (Lin et al., 2020a; Agarwal et al., 2023; Mukherjee et al., 2023), with ORCA (Mukherjee et al., 2023) reaching parity performance with ChatGPT (OpenAI, 2023) by distilling the reasoning traces from GPT4 (OpenAI, 2023). Distinct from this line of work, RLGF does not aim to replicate the guidance policy. Rather, our objective is to leverage generation traces derived from a guide policy to condense the search space for RL algorithms. More importantly, our goal goes beyond imitation of the guidance policy and focuses on algorithms that better optimize a reward with guidance policy feedback.

3 Preliminaries

Text generation with LLMs can be viewed as a structured prediction problem, consisting of an input space $\mathcal{X}$, an output space $\mathcal{Y}$ and non-negative loss function $\ell(x, \hat{y}, y*) \mapsto \mathbb{R}^{\geq 0}$ such that the loss function ℓ represents how close $\hat{y}$ is to the ground truth y^* given the input x. We are provided with a training set of N labeled input-output pairs $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$ drawn from some unknown distribution over $\mathcal{X} \times \mathcal{Y}$. The goal is to learn a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ that minimizes the loss function ℓ with respect to $\mathcal{D}$. We adopt the approach of solving the text generation structured prediction problems using sequential decision-making as formalized in learning-to-search (L2S) (Daumé et al., 2009; Collins & Roark, 2004; Ratnaparkhi, 1996).

We view our L2S problem as a token-level finite-horizon MDP $\langle \mathcal{S}, \mathcal{A}, P, R, H, \mu \rangle$ using a finite vocabulary $\mathcal{V}$. We are given a labeled dataset $\mathcal{D} = \left\{(x^i, y^i)\right\}_{i=1}^N$ of N samples, where x^i is a prompt text and y^i is the target text generation. We define $\mu \in \Delta(\mathcal{D})$ as the initial distribution over prompts in the dataset, and the action space $\mathcal{A}$ as the set of tokens in our vocabulary $\mathcal{V}$. The state space $\mathcal{S} = \cup_{h=1,\cdots,H} \mathcal{V}^h$ is the set of all possible token sequences and a state $s_h \in \mathcal{S}$ is the prompt x and previously generated tokens $(a_0, a_1, \ldots, a_{h-1})$, i.e., $s_h = (x, a_0, a_1, \ldots, a_{h-1})$. The transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a deterministic known transition function that appends the next action a_h to the state s_{h+1} The time horizon $H \in \mathbb{Z}_+$ is the maximum generation length. Finally, $R : \mathcal{S} \to \mathbb{R}$ is the reward function such as the task evaluation metric.

Let d_h^π represent the state distribution of visiting a state at time h. Let $d^\pi = \frac{1}{H}\sum_{h=0}^H d_h^\pi$ be the average visitation if we follow π for H steps in a trajectory. With an LLM policy π, we define the value function and Q-function as $V_h^\pi(s) = \mathbb{E}_\pi[\sum_{h'=h}^H R(s_{h'})|s_h = s]$ and $Q_h^\pi(s,a) = R(s) + \mathbb{E}_{s' \sim P(\cdot|s,a)}[V_{h+1}^\pi(s')]$ respectively. Finally, we define the advantage function for an LLM policy π as $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$.

Guide policy π^g In our setting, we additionally assume access to an LLM guide policy π^g that can assist our policy π. The guide policy can be used to alter the initial state distribution μ and to compute the advantage function $A^{\pi^g}(s,a)$. In this work, π^g is a supervised fine-tuned (SFT) model on the downstream task and generate feedback from π^g with a more effective decoding strategy like nucleus sampling (Holtzman et al., 2019). Note, RLGF treats π^g as a query-able, black-box model that we cannot update. This allows for π^g to be any black-box model such as GPT4 or a human-expert.

4 Reinforcement Learning from Guided Feedback

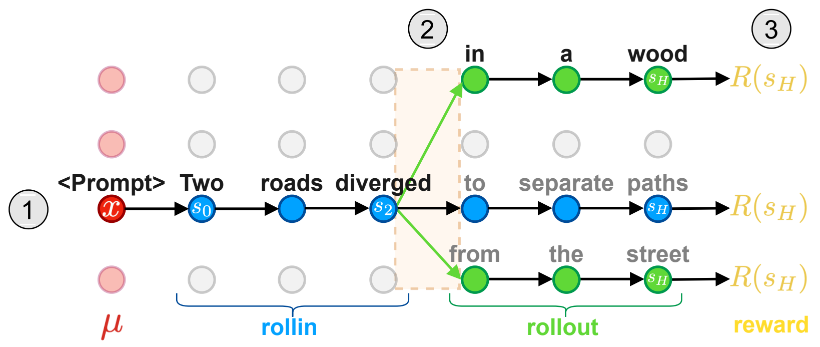

Figure 1: RLGF's main mechanism of incorporating guidance through interactions between two LLMs: rollin and rollout policies. (1) the rollin policy generates a trajectory. (2) the rollout policy restarts to a sampled point in the generation (i.e. s_2) and completes the generation. (3) the rollout policy receives a score (i.e. reward) for the generation.

Unlike other tasks studied in RL, structured prediction problems such as text generation, have two key properties: a deterministic transition function and a policy's ability to restart to any state. Because our transition function is the set of previously generated tokens, we can easily alter the words in the generation (add, remove or swap), and restart our policy π_θ to any point of the generation.

Restarts allow us to execute rollin and rollout policies as seen in Figure 1. The rollin policy is used to generate sequences that the rollout policy evaluates. Specifically, we sample a prompt x and target sentence y from our initial distribution μ. We then generate an entire trajectory using our rollin policy starting from the sampled prompt. We combine the state-action pairs from the collected rollin trajectory with the initial state distribution – creating a modified initial state for the rollout policy. The rollout policy samples a state along the rollin generation, restarts to this state and performs a one-step deviation action. The rollout policy then completes the generation and collects a reward. The rollin and rollout policies can be our LLM policy π_θ, guide policy π^g or a mixture that interpolates between the two. Depending on the choice of rollin and rollout policies, we invoke different algorithms.

PPO: Rollin π_θ and Rollout π_θ Under this schematic, notice how when both the rollin and rollout policies are our current LLM policy π_θ that is being fine-tuned, the resulting RL algorithm is PPO. That is, we would be collecting generations from a single LLM. This configuration does not take advantage of the ability to modify the initial state distribution nor the availability of a guide policy π^g.

Algorithm 1 PPO^{++}

```
1: Input: $\pi_\theta$, guide $\pi^g$, iterations $T$, mixing parameter $\beta \in [0,1]$, dataset $\mathcal{D} = \left\{(x^i, y^i)\right\}_{i=1}^N$
2: for $t \in [T]$ do
3:     Rollin with $(s,a) \sim \beta d^{\pi^g} + (1-\beta) d^{\pi_\theta^t}$ starting from $x \sim \mathcal{D}$
4:     Rollout with $\pi_\theta^t$ to collect trajectories
5:     Update $V_\phi^{\pi_\theta^t}$ with trajectories and compute advantage estimates $A^{\pi_\theta^t}$
6:     Update $\pi_\theta$ using PPO loss with $A^{\pi_\theta^t}$
7: return $\pi_\theta$
```

PPO^{++}: Rollin π^g and Rollout π_θ The second scheme we consider is rollin with our guide policy π^g and rollout with our LLM policy π_θ. This strategy is motivated from a popular Approximate Policy Iteration algorithm (Bertsekas, 2011): Conservative Policy Iteration (CPI) (Kakade & Langford, 2002). CPI proposes to use a diverse initial state distribution to address the exploration issue in PG methods. Particularly, it proposes to use an initial state distribution that covers some high-quality policy distribution. The first key idea of PPO^{++} is to take advantage of a guide policy π^g to provide an enlarged initial state distribution – so that the rollout policy, π_θ, can visit diverse and relevant states it would otherwise not visit. The second key idea of PPO^{++} is using a mixture policy with state distribution $\beta d^{\pi^g} + (1-\beta) d^{\pi_\theta}$, for rollin (see Algorithm 1 Line 3). This ensures that with probability $(1-\beta)$, PPO^{++} is executing the default PPO update, making sure PPO^{++} never underperforms PPO.

Algorithm 2 AggreVaTeD

```
1: Input: $\pi_\theta$, guide $\pi^g$, iterations $T$, mixing parameter $\beta \in [0,1]$, dataset $\mathcal{D} = \left\{(x^i, y^i)\right\}_{i=1}^N$
2: for $t \in [T]$ do
3:     Rollin with $(s,a) \sim (1-\beta) d^{\pi_\theta^t} + \beta d^{\pi^g}$ starting from $x \sim \mathcal{D}$
4:     Rollout with $\pi^g$ to collect trajectories
5:     Update $V_\phi^{\pi^g}$ with trajectories and compute advantage estimates $A^{\pi^g}$
6:     Update $\pi_\theta$ using PPO loss with $A^{\pi^g}$
7: return $\pi_\theta$
```

AggreVaTeD: Rollin π_θ and Rollout π^g The next scheme performs rollin with our LLM policy π_θ and rollout with our guide policy π^g – the opposite of PPO^{++}. This scheme is an interactive imitation learning algorithm, AggreVaTeD (Sun et al., 2017), a differentiable policy gradient version of AggreVaTe (Aggregate Values to Imitate (Ross & Bagnell, 2014)) as seen in Algorithm 2. AggreVaTeD is an API algorithm similar to CPI and also uses a mixture policy with state distribution $\beta d^{\pi^g} + (1-\beta) d^{\pi_\theta}$ for rollin. This algorithm first generates rollins with the mixture policy to collect sequences. Then AggreVaTeD generates rollouts with the guide policy and evaluates the quality of the generated rollouts. It then uses the rollouts to train a value network $V_\phi^{\pi^g}$ that measures the reward-to-go of π^g, which in turn is used to construct the advantage of π^g: A^{π^g}. With this advantage A^{π^g}, AggreVaTeDupdates the policy like PPO. Intuitively, the algorithm aims to learn the policy $\arg\max_a A^{\pi^g}(s,a)$. Rolling out with π^g ensures that the LLM policy π_θ can be *at least* as good as or better than the guide policy π^g.

Algorithm 3 D^{2}LOLS

```
1: Input: $\pi_\theta$, guide $\pi^g$, iterations $T$, dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$
2: Run $\pi_\theta^1 = \texttt{AggreVaTeD}(\pi_\theta, \pi^g, \alpha T, \beta_1, \mathcal{D})$
3: Run $\pi_\theta^2 = \texttt{PPO}^{++}(\pi_\theta^1, \pi^g, (1-\alpha)T, \beta_2, \mathcal{D})$
4: return $\pi_\theta^2$
```

D^{2}LOLS: combines PPO^{++} and AggreVaTeD Given the previous approaches of interaction, we can come up with multiple ways to combine PPO, PPO^{++}, and AggreVaTeD. In Algorithm 3, we present Direct and Differentiable Locally Optimal Learning to Search (D^{2}LOLS), which is a simple approach to combine the previous methods. D^{2}LOLS is a differentiable policy gradient version of Locally Optimal Learning to Search (LOLS)(Chang et al., 2015) and addresses limitations of how LOLS combines PPO, PPO^{++}, and AggreVaTeD. The original formulation of LOLS requires computing cost-sensitive classification similar to AggreVaTe; instead we take inspiration from AggreVaTeD's differentiable approach to develop a differentiable version of LOLS. Furthermore, LOLS (Algorithm 4) has a mixing probability parameter α which directly merges the advantage function between PPO and AggreVaTeD, leading to theoretical issues. D^{2}LOLS removes this mixing probability and replaces it with a mixing time variable α that decides how many iterations to perform AggreVaTeD before switching to PPO^{++}. This simple strategy fixes LOLS's issue arising from interweaving guidance.

5 Theoretical Justification

In this section, we provide theoretical justification for various rollin and rollout schemes mentioned in Section 4. Each algorithmic scheme takes advantage of a guide policy π^g, the ability to restart the policy to any state, and access to the reward signal. Our theoretical justification are derived from the original algorithms that each method has built upon.

Interactive Imitation Learning: AggreVaTeD In our interactive IL setting, we assume access to the ground truth reward and to a guide policy π^g that may not necessarily be an expert policy $\pi^\star$ (i.e. optimal at the task). Our AggreVaTeD (Algorithm 2) implementation is a modification of the original AggreVaTeD (Sun et al., 2017) to incorporate a PPO policy gradient loss. The overall idea is to perform policy gradient updates on the loss function $\ell_t(\pi) := \mathbb{E}_{s\sim d^{\pi^t}} \mathbb{E}_{a\sim\pi(\cdot|s)}[A^{\pi^g}(s,a)]$, where π^t is our latest learned policy. We can define the average-regret and best policy performance in our policy class over T-iterations as:

$$\epsilon_{\text{regret}} = \frac{1}{T}\left(-\sum_{t=0}^{T} \ell_t(\pi^t) + \max_{\pi\in\Pi} \sum_{t=0}^{T} \ell_t(\pi)\right) \quad \epsilon_{\text{class}} = \max_{\pi\in\Pi} \frac{1}{T}\sum_{t=0}^{T} \mathbb{E}_{s\sim d^{\pi^t}}\left[A^{\pi^g}(s,\pi(s))\right].$$

If the gradient update procedure achieves no-regret, i.e., $\epsilon_{\text{regret}} \to 0$ as $T \to \infty$, AggreVaTeD achieves the following guarantee; there exists $t \in [T]$, such that:

$$V^{\pi^t} \geq V^{\pi^g} + H\epsilon_{\text{class}}.$$

When the guide policy is included in our policy class $\pi^g \in \Pi$, e.g., when our policy π_θ and our guide π^g have the same GPT2 model architecture, then our ϵ_{class} term is guaranteed to be non-negative. Furthermore, this term is positive when π^g is not globally optimal with respect to its advantage function (i.e., $\max_a A^{\pi^g}(s,a)$ can be positive). Thus when $\epsilon_{\text{regret}} \to 0$ (i.e., no-regret), AggreVaTeD guarantees to learn a policy π_t that outperforms the guide policy by a margin. This was originally confirmed empirically in Sun et al. (2017) and is also confirmed in our experiments. With our SFT model with nucleus sampling as π^g, AggreVaTeD learns a policy π^t outperforming π^g.

Reinforcement Learning with better restart distribution: PPO^{++} Although AggreVaTeD is capable of outperforming π^g, it is an imitation learning algorithm, meaning by design, its performance is limited by the performance of π^g. In contrast, RL has the potential to learn the near optimal policy, but popular RL approaches suffer from a lack of exploration. We propose to leverage rollin's with the guide policy to overcome RL's exploration issues. PPO^{++} Algorithm 1 implements this idea using a PPO loss. We can interpret the rollin policy distribution with the guide policy, as a restart distribution that alters the initial distribution of our policy, i.e., $\mu_{\text{mix}} := (1-\beta)\mu + \beta d^{\pi^g}$, where recall $\mu \in \Delta(\mathcal{D})$ is the original initial state distribution over our data.

Policy gradient theory (Kakade & Langford, 2002; Bagnell et al., 2003; Agarwal et al., 2019; 2021) ensures that as long as a near optimal policy is covered by the restart distribution, we can learn to perform as well as the near optimal policy. More formally, consider the special case where $\beta = 1/2$, and $\pi^\star$ is the globally optimal policy; and assume that at some iteration t one-step local improvement over π^t is small, i.e., $\mathbb{E}_{s,a\sim d^{\pi^t}_{\mu_{\text{mix}}}}\left[\max_a A^{\pi^t}(s,a)\right] \leq \epsilon$, then with some small ϵ we have:

$$V^{\pi^t} \geq V^{\pi^\star} - O\left(H^2 \max_s \left(\frac{d^{\pi^\star}(s)}{d^{\pi^g}(s)}\right)\epsilon\right)$$

We refer readers to the proof of theorem 6.2 in Kakade & Langford (2002). Note that compared to the result from AggreVaTeD, we are able to compare against the globally optimal policy $\pi^\star$ under the condition that π^g's state distribution covers $\pi^\star$'s state distribution (i.e., the guide policy has a good sense of what states $\pi^\star$ will likely visit). In our experiments, we mainly use a SFT model with nucleus sampling as our guide policy π^g. While we do not expect the SFT policy π^g is as good as the optimal $\pi^\star$, it is reasonable to expect that d^{π^g} provides coverage to $d^{\pi^\star}$. Our experiments verify that restarting based on states from d^{π^g} improves the performance of PPO.

Combine Reinforcement Learning and Imitation Learning: D^{2}LOLS D^{2}LOLS is the simplest approach to combine AggreVaTeD and PPO^{++}. This algorithm runs AggreVaTeD for a fixed period of time and then PPO^{++} for the remaining time. If our policy gradient algorithm is Trust-region policy optimization (TRPO) [1] (Schulman et al., 2015) or CPI (Kakade & Langford, 2002), then our algorithm has a guaranteed monotonic policy improvement. This means that upon convergence, we achieve two properties: (1) our learned policy is at least as good or better than the guide policy π^g, (2) our policy is locally optimal, i.e., the local one-step improvement, $\mathbb{E}_{s,a\sim d^{\pi}_{\mu_{mix}}}[\max_a A^{\pi}(s,a)]$, has to be small (otherwise TRPO and CPI can keep improving).

There exist several algorithms in the literature that combine RL and IL (Cheng et al., 2018; Sun et al., 2018; Chang et al., 2015; Rajeswaran et al., 2017; Nair et al., 2018). The key difference between D^{2}LOLS and LOLS is how PPO^{++} and AggreVaTeD is combined. LOLS uses a mixing probability α to combine our π_θ and the guide policy π^g advantage function $\alpha A^{\pi^t_\theta} + (1-\alpha)A^{\pi^g}(s,a)$; whereas D^{2}LOLS uses a mixing time parameter α to decide when to switch from doing AggreVaTeD to PPO^{++} for the remainder of training. LOLS can achieve the property of outperforming better than π^g and also being locally optimal, but *only under* the assumption that the following gap is small:

$$\forall \pi : \left| \mathbb{E}_{s\sim d^\pi}\left[\max_a A^{\pi^g}(s,a) + \max_a A^{\pi}(s,a)\right] - \mathbb{E}_{s\sim d^\pi} \max_a \left[A^{\pi^g}(s,a) + A^{\pi}(s,a)\right] \right| \leq \varepsilon,$$

with some small ε. However, such a gap can exist in practice and does not vanish even with enough training data. Intuitively this gap is non-trivial when the one-step improvement over π contradicts with the one-step improvement over π^g. The simplest approach D^{2}LOLS works the best, and achieves the guarantee that LOLS aimed for without the additional assumption of the above gap being small.

6 Experiments

We perform all of our experiments using a modified PPO objective J_{ppo} (Ouyang et al., 2022; Wu et al., 2016). This objective combines the original PPO objective with a maximum-likelihood estimation (MLE) objective of the ground-truth dataset's $\mathcal{D}$ references:

$$J_{ppo}(\pi) = \mathbb{E}_{(s,a)\sim\pi_\theta}\Big[R(s) - \lambda \text{KL}(\pi_\theta(a|s)||\pi_0(a|s))\Big] + \eta \mathbb{E}_{(s,a)\sim\mathcal{D}}\Big[\log \pi_\theta(a|s)\Big],$$

where λ is the KL coefficient and η is the MLE coefficient. For all of our proposed RLGF algorithms discussed in section 4 we consider setting π^g to the supervised fine-tuned model (SFT) with nucleus

[1] in our experiments, instead of using TRPO, we use PPO – a scalable version of TRPO that is more suitable for high-dimensional problems. However we emphasize the TRPO and PPO use the same principle for policy optimization: make conservative policy update (Kakade & Langford, 2002) to ensure monotonic improvement.

Algorithms	**IMDB Sentiment** *Semantic and Fluency Metrics*			**CommonGen** *Lexical and Semantic Metrics*			
	Sentiment Score (↑)	Perplexity (↓)	Output-Perplexity (↓)	Bleu-4 (↑)	BERTScore (↑)	**CIDEr-D** (↑)	**SPICE** (↑)
Zero-Shot	0.48 ± 0.00	32.55 ± 0.00	5.64 ± 0.00	0.16	0.93	1.10	0.26
`SFT`	0.55 ± 0.00	35.67 ± 0.00	6.19 ± 0.00	0.22	0.95	1.43	0.31
`SFT+PPO`	0.97 ± 0.01	44.92 ± 1.78	3.17 ± 0.62	0.26	0.95	1.65	0.32
`SFT+PPO`$^{++}$	0.97 ± 0.01	44.83 ± 2.10	3.34 ± 0.80	0.27	0.95	1.68	0.32
`SFT+AggreVaTeD`	0.95 ± 0.03	52.56 ± 5.38	5.04 ± 2.30	0.27	0.95	1.65	0.32
`SFT+LOLS`	0.93 ± 0.05	53.30 ± 16.70	3.44 ± 4.96	0.26	0.95	1.66	0.32
`SFT+D`^{2}`LOLS`	0.97 ± 0.00	43.88 ± 2.37	2.92 ± 0.13	0.27	0.95	1.69	0.33

Table 1: **IMDB and CommonGen Results:** We compute the mean and standard deviation over 3 seeds for the IMDB task and compute 1 seed for the CommonGen task. For our reward function each task we use the bold metric(s). The zero-shot model is the performance of the pretrained model used for IMDB and CommonGen, GPT-2 and T5 respectively. `SFT+Alg` indicates running `Alg` after supervised finetuning. `SFT+nucleus` is used as our guide policy π^g for all experiments.

sampling for decoding (i.e., $\pi^g =$`SFT+nucleus`). We treat `SFT+nucleus` as a black-box model that we can only query for text generation and do not perform updates to it. By using `SFT+nucleus` as our guide policy, we run all of our experiments under the exact same conditions as those of RLHF. Note, RLHF already requires keeping `SFT` to compute the KL constraint, $KL(\pi_\theta || \pi_0)$, in J_{ppo}.

Task Details In our experiments, *perplexity* measures how likely our learned model, π_θ, is to generate the references in the task dataset, whereas *output perplexity* computes how likely a general LLM (e.g. GPTJ) is to generate the generations from our learned policy, π_θ. Both perplexity metrics have been reported as a measure of fluency (Fedus et al., 2018; Ramamurthy et al., 2022).

We perform experiments on three tasks. IMDB is the first task and the objective is to generate fluent and positively sentiment-ed text continuations for IMDB (Maas et al., 2011) movie reviews prompts. We use a sentiment classifier (Sanh et al., 2019) as our reward function that is trained on review texts and sentiment labels from the dataset, which then provides sentiment scores indicating how positive a given piece of text is. For training supervised `SFT` baselines, we consider only the examples with positive labels. We chose `GPT2` (Radford et al., 2019) as the base language model (LM) for this task. We evaluate all algorithms on three metrics: sentiment reward score, perplexity, and output-perplexity.

Next, we consider CommonGen (Lin et al., 2020b), a challenging constrained, text generation task that tests the ability of generative common sense reasoning. We optimize the SPIDER (Liu et al., 2017) reward function, a weighted combination of the CIDEr-D and SPICE metric. We chose `T5-base` (Raffel et al., 2020) as our base LLM and prefixed each concept set input with: "generate a sentence with:". We evaluate on four metrics: BLEU (Papineni et al., 2002), CIDEr-D (Vedantam et al., 2015), SPICE (Anderson et al., 2016), and BERTScore (Zhang et al., 2019a).

The final task we consider is Reddit TL;DR summarization dataset (Völske et al., 2017) where the objective is to generated summaries. We use the filtered dataset with additional human preference data used in Stiennon et al. (2020). The base LLM that we use for this task is GPT-J (Wang & Komatsuzaki, 2021) and we train all models using LoRA adapters(Hu et al., 2021). We evaluate all algorithms on 5 metrics: reward score, perplexity, output-perplexity, win rate and Rouge (Lin, 2004). For win rate, we use the open source Llama2-13B-chat (Touvron et al., 2023) model as our evaluator model. We compare all algorithm generations to the preferred summary references. Refer to Appendix C.2, for the exact Win Rate prompt, example evaluations and implementation details.

6.1 Experimental Results

RLGF vs. RLHF Performance Table 1 and Table 2 compares all of the RLGF algorithms proposed in Section 4 against standard RLHF algorithms and baselines. For all tasks, our π^g is `SFT` which is sub-optimal, performing worse than all RL based algorithms across most lexical and semantic metrics. Utilizing this π^g, for IMDB, `SFT+D`^{2}`LOLS` and `PPO`$^{++}$ outperform `PPO`, and for CommonGen, `D`^{2}`LOLS` outperforms `PPO` . Finally, for TL;DR summarization we see that `PPO`$^{++}$ performs better than `PPO` as well as a competitive baseline, `Best-of-N` (Dubois et al., 2023).

Algorithms	TL;DR Summarization *Semantic and Fluency Metrics*						
	RM Score (↑)	Perplexity (↓)	Output-Perplexity (↓)	Win Rate (↑)	Rouge 1 (↑)	Rouge 2 (↑)	RougeL (↑)
Zero-Shot	1.57	14.07	11.51	44.12%	0.27	0.07	0.18
SFT	5.68	14.09	12.81	44.29%	0.34	0.25	0.25
Best-of-N$(N = 8)$	5.98	14.09	12.86	47.60%	0.36	0.13	0.27
SFT+PPO	6.01	15.05	17.67	54.25%	0.35	0.13	0.27
SFT+PPO^{++}	6.11	14.53	16.15	55.01%	0.36	0.14	0.27
SFT+AggreVaTeD	5.93	14.69	16.41	48.98%	0.36	0.15	0.29

Table 2: **TL;DR Summarization Results:** We report the mean over 1 seed. Our RM Score is under our trained preference reward model and the Win Rate is evaluated by Llama2-13B-Chat. We use SFT+nucleus as π^g.

Supporting our justification from Section 5, AggreVaTeD improves beyond our guide policy, providing an alternative as a warm-starting methodology to warm-starting with SFT. As shown by Table 7, we see that warm-starting with AggreVaTeD leads to higher performance on IMDB than warm-starting with SFT, a popular learning strategy when performing RL for language (Stiennon et al., 2020; Ouyang et al., 2022). PPO^{++}, on the other hand, is better than or competitive to our RL baseline demonstrating a simple, yet powerful alternative to PPO as the RL procedure. Even in practice, we observe the benefit of restarting from an initial state distribution that better covers an optimal policy's state distribution. The combination of these two, D^{2}LOLS, achieves the best of both worlds and fully leverages the capabilities of utilizing a guide policy.

Reward Optimization Tradeoff In Figure 2 we evaluate how well RLGF algorithms trade-off optimizing the reward while minimizing the perplexity and kl-constriant $\sqrt{\text{KL}}$. For both plots, the top right corner indicates the policy has both high reward and low perplexity and low divergence from π_0. For each algorithm we plot 5 checkpoints ranging from 20 to 100 iterations.PPO^{++} mostly matches or has higher reward than PPO while maintaining a lower perplexity. Separately, AggreVaTeD trade-offs reward for perplexity, and has comparable reward scores as PPO while drastically reducing its perplexity. For the kl-constraints plot on the left of Figure 2 we see that although PPO has a set of points with high reward, most of these points also have high KL divergences. Whereas, a subset of PPO^{++} matches or has higher reward than PPO while having a lower kl-constraint.

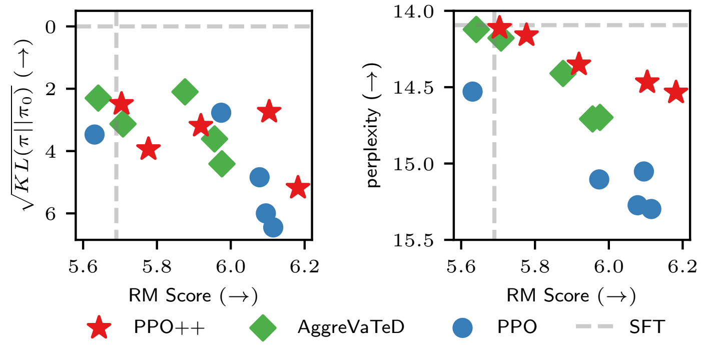

Figure 2: We investigate the reward optimization, kl-constriant, and fluency trade-off in our TL;DR summarization task. The dashed line represents our SFT policy's performance across each metric. Both PPO^{++} and AggreVaTeD learn a policy that has a better trade-off than PPO.

RLGF Performance on Difficult Prompts Our evaluation was carried out on the CommonGen task where we categorized the prompts based on their difficulty level. For CommonGen, we classify the prompts into *easy* and *hard* based on the number of unseen concepts in the prompt. Specifically, we categorized prompts with 3 concepts as easy and more than 3 concepts as hard. Figure 3 presents a comparison of scores for different algorithms grouped by prompt difficulty. The results reveal a notable performance gap between easy and hard prompts for algorithms such as SFT and PPO,

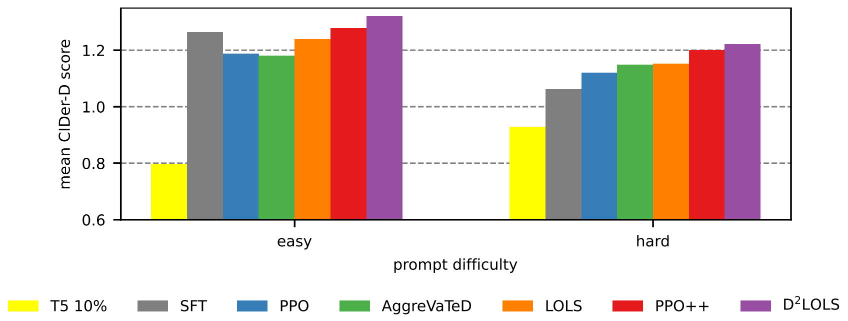

Figure 3: Comparison of CIDer-D scores grouped by prompt difficulty on CommonGen. The performance gap between easy and hard prompts is evident for SFT, and PPO^{++}, while our proposed algorithms AggreVaTeD, LOLS and D^{2}LOLS exhibit a significantly smaller gap, showcasing their effectiveness on challenging prompts.

whereas our proposed algorithms PPO^{++}, AggreVaTeD, LOLS and D^{2}LOLS exhibit a significantly smaller gap, with D^{2}LOLS having the least gap. In other words, even on challenging prompts, our interactive algorithms produce better text continuations. See Appendix E for example generations.

MLE and KL coefficient Sensitivity We test the sensitivity of PPO and RLGF algorithms to two regularization hyperparameters in the J_{ppo} objective, namely the KL coefficient, λ, and the MLE coefficient, η. The left 2 plots in Figure 4 show the reward and perplexity when we keep η fixed and vary λ while the right 2 show the performance when we keep λ fixed and vary η. All RL algorithms are robust to varying KL coefficients. We observe much more instability when relaxing our MLE regularization with PPO and RLGF algorithm's perplexities blowing up.

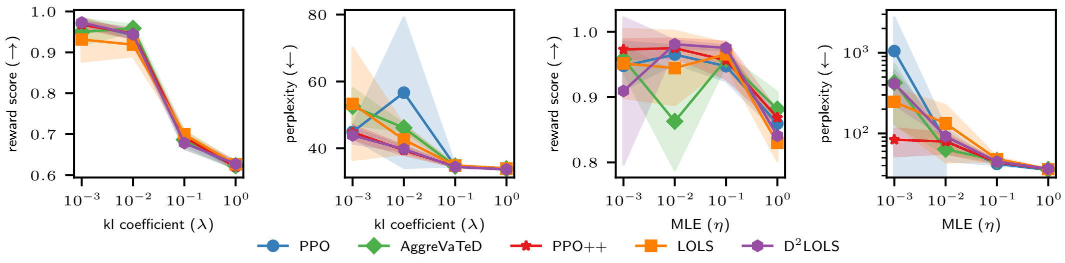

Figure 4: J_{ppo} KL coefficient (λ) and MLE coefficient (η) ablation. We show the sensitivity of PPO and RLGF algorithms to each regularization term in the objective. Note that all RL algorithms are robust to changes in KL coefficient with relatively minor changes in the Perplexity while being more sensitive to changes in MLE objective (Right) with blowups in the perplexity.

7 Conclusion and Future Work

We presented a unifying framework of incorporating a guide policy to enhance reinforcement learning for natural language generation. Through theoretical justification and experimental validation, we demonstrate that our RLGF framework can outperform PPO for fine-tuning LLMs. Our proposed algorithms PPO^{++} and D^{2}LOLS only require black-box access to the guide policy and are conceptually simple and easy to implement based on PPO. While in our experiment, we demonstrate that supervised fine-tuned models with standard decoding strategies is a good candidate of the guide policy, our framework is general enough to leverage any large LLMs as the guide policy, including those that are not open-sourced. Finally, RLGF's contributions to the broader large language model literature is complementary to model enhancements, dataset improvements, and prompting discoveries such as in-context prompting. We leave it to exciting future work to test the full capabilities of bootstrapping the state-of-the-art advancements in each research direction with RLGF to improve reinforcement learning for natural language generation.

REFERENCES

Alekh Agarwal, Nan Jiang, and Sham M Kakade. Reinforcement learning: Theory and algorithms. 2019.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In *Conference on Learning Theory*, pp. 64–66. PMLR, 2020.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift. *Journal of Machine Learning Research*, 22(1):4431–4506, 2021.

Rishabh Agarwal, Nino Vieillard, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*, 2023.

Prithviraj Ammanabrolu and Mark O Riedl. Playing text-adventure games with graph-based deep reinforcement learning. *arXiv preprint arXiv:1812.01628*, 2018.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic Propositional Image Caption Evaluation. In *Proceedings of European Conference on Computer Vision*, 2016.

James Bagnell, Sham M Kakade, Jeff Schneider, and Andrew Ng. Policy Search by Dynamic Programming. *In Proceedings of Neural Information Processing Systems*, 2003.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Michiel Bakker, Martin Chadwick, Hannah Sheahan, Michael Tessler, Lucy Campbell-Gillingham, Jan Balaguer, Nat McAleese, Amelia Glaese, John Aslanides, Matt Botvinick, et al. Fine-tuning language models to find agreement among humans with diverse preferences. *Advances in Neural Information Processing Systems*, 35:38176–38189, 2022.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *In Proceedings of Neural Information Processing Systems*, 2015.

Dimitri P Bertsekas. Approximate Policy Iteration: A Survey and Some New Methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of Artificial General Intelligence: Early Experiments with Gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

CarperAI. https://github.com/carperai/trlx, 2023.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to Search Better than your Teacher. In *Proceedings of International Conference on Machine Learning*, 2015.

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R Bowman, Kyunghyun Cho, and Ethan Perez. Improving code generation by training with natural language feedback. *arXiv preprint arXiv:2303.16749*, 2023.

Ching-An Cheng, Xinyan Yan, Nolan Wagener, and Byron Boots. Fast Policy Learning through Imitation and Reinforcement. *arXiv preprint arXiv:1805.10413*, 2018.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 111–118, 2004.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*, pp. 41–75. Springer, 2019.

Hal Daumé, John Langford, and Daniel Marcu. Search-based Structured Prediction. *Machine learning*, 75:297–325, 2009.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

Daniel Duckworth, Arvind Neelakantan, Ben Goodrich, Lukasz Kaiser, and Samy Bengio. Parallel Scheduled Sampling. *arXiv preprint arXiv:1906.04331*, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Github. https://github.com/features/copilot, 2023. Accessed: 2023-May-13.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215*, 2023.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. Differentiable Scheduled Sampling for Credit Assignment. *arXiv preprint arXiv:1704.06970*, 2017.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. News Summarization and Evaluation in the Era of Gpt-3. *arXiv preprint arXiv:2209.12356*, 2022.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, pp. 1884. NIH Public Access, 2018.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded Language Learning in a Simulated 3D World. *arXiv preprint arXiv:1706.06551*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge ina neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proceedings of Annual Conference on Robot Learning*, 2022.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Sham Kakade and John Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of International Conference on Machine Learning*, 2002.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to controlled text generation. *arXiv preprint arXiv:2012.11635*, 2020.

Khan Academy. https://blog.khanacademy.org/harnessing-ai-so-that-all-students-benefit-a-nonprofit-approach-for-equal-access/, 2023. Accessed: 2023-May-14.

Samuel Kiegeland and Julia Kreutzer. Revisiting the weaknesses of reinforcement learning for neural machine translation. *arXiv preprint arXiv:2106.08942*, 2021.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. 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.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. Can neural machine translation be improved with user feedback? *arXiv preprint arXiv:1804.05958*, 2018a.

Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. *arXiv preprint arXiv:1805.10627*, 2018b.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. SEARNN: Training RNNs with Global-local losses. *arXiv preprint arXiv:1706.04499*, 2017.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023a.

Peter Lee, Sebastien Bubeck, and Joseph Petro. Benefits, Limits, and Risks of GPT-4 as an AI Chatbot for Medicine. *New England Journal of Medicine*, 388(13):1233–1239, 2023b.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2016.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. *arXiv preprint arXiv:2009.07253*, 2020a.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In *Findings of Association for Computational Linguistics: EMNLP*, 2020b.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. Improved Image Captioning via Policy Gradient Optimization of Spider. In *Proceedings of International Conference on Computer Vision*, 2017.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In *Proceedings of Annual Meeting of the Association for Computational Linguistics: Human language technologies*, 2011.

Tsvetomila Mihaylova and André FT Martins. Scheduled Sampling for Transformers. *arXiv preprint arXiv:1906.07651*, 2019.

Dipendra Misra, John Langford, and Yoav Artzi. Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2017.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language Understanding for Text-based Games using Deep Reinforcement Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2015.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. *arXiv preprint arXiv:1707.07402*, 2017.

OpenAI. https://openai.com/blog/chatgpt, 2023.

OpenAI. Gpt-4 technical report, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback. *In Proceedings of Neural Information Processing Systems*, 2022.

Richard Yuanzhe Pang and He He. Text Generation by Learning from Demonstrations. In *Proceedings of International Conference on Learning Representations*, 2021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of Association for Computational Linguistics*, 2002.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is Reinforcement Learning (Not) for Natural Language Processing?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization. *arXiv preprint arXiv:2210.01241*, 2022.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Conference on empirical methods in natural language processing*, 1996.

Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. Deep Reinforcement Learning-Based Image Captioning With Embedding Reward. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Paul Roit, Johan Ferret, Lior Shani, Roee Aharoni, Geoffrey Cideron, Robert Dadashi, Matthieu Geist, Sertan Girgin, Léonard Hussenot, Orgad Keller, et al. Factually consistent summarization via reinforcement learning with textual entailment feedback. *arXiv preprint arXiv:2306.00186*, 2023.

Stephane Ross and J Andrew Bagnell. Reinforcement and Imitation Learning via Interactive No-regret Learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-regret Online Learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2011.

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*, 2023.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization). In *Proceedings of International Conference on Machine Learning*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*, 2015.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Artem Sokolov, Stefan Riezler, and Tanguy Urvoy. Bandit structured prediction for learning from partial feedback in statistical machine translation. *arXiv preprint arXiv:1601.04468*, 2016.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to Summarize with Human Feedback. *In Proceedings of Neural Information Processing Systems*, 2020.

Theodore R Sumers, Mark K Ho, Robert D Hawkins, Karthik Narasimhan, and Thomas L Griffiths. Learning rewards from linguistic feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6002–6010, 2021.

Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply Aggrevated: Differentiable Imitation Learning for Sequential Prediction. In *Proceedings of International Conference on Machine Learning*, 2017.

Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated Horizon Policy Search: Combining Reinforcement Learning & Imitation Learning. *arXiv preprint arXiv:1805.11240*, 2018.

Arash Tavakoli, Vitaly Levdik, Riashat Islam, Christopher M Smith, and Petar Kormushev. Exploring restart distributions. *arXiv preprint arXiv:1811.11298*, 2018.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based Image Description Evaluation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4566–4575, 2015.

Arun Venkatraman, Martial Hebert, and J Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pp. 59–63, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4508. URL https://aclanthology.org/W17-4508.

Ben Wang and Aran Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

Sam Wiseman and Alexander M Rush. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2016.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. Rlcd: Reinforcement learning from contrast distillation for language model alignment. *arXiv preprint arXiv:2307.12950*, 2023.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating Text Generation with Bert. *arXiv preprint arXiv:1904.09675*, 2019a.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. *arXiv preprint arXiv:1906.02448*, 2019b.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. Greaselm: Graph reasoning enhanced language models. In *Proceedings of International Conference on Learning Representations*, 2022.

Xingxing Zhang and Mirella Lapata. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2017.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning Language Models from Human Preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Additional Related Work

LLM Alignment Using RLHF is one idea of aligning LLM with human preferences. The RLHF objective incorporates a KL constraint and is equivalent to minimizing the reverse KL between KL-control distribution and the learner. Minimizing some divergence between policy used for the KL-control and learner policy has been proposed for LLM alignment. (Korbak et al., 2022; Khalifa et al., 2020; Go et al., 2023) propose alignment ideas the attempt to minimize various divergence inspired from maximize entropy RL (Haarnoja et al., 2017; 2018) and Distributional Policy Gradient (DPG) (Barth-Maron et al., 2018). Depending on the chosen divergence, the desired policy behavior may be easy or hard to obtain. Another collection ideas for alignment focus on aspects of the supervised learning data, for example currating the collected data (Zhou et al., 2023; Chung et al., 2022).

Restart Distribution On-policy RL algorithms are not able to take advantage of past visited states. But incorporating the ability to reset to any arbitrary state allows on-policy methods to create new states from past visited states (Tavakoli et al., 2018). The core of the idea is to use past visited states to modify the initial state distribution. Our work introduces $\texttt{PPO}^{++}$ which is an algorithm that has no prior over past visited states but (Tavakoli et al., 2018) considers incorporating priories to help decide how to prioritize past visited states to incorporate into the initial state distribution. (Agarwal et al., 2020) showed theoretically that the initial state distribution helps with exploration. Modifying the initial state distribution using restart has seen success in Montezuma Revenge Atari 2600 (a hard exploration problem) and Atari 2600 games more broadly(Popov et al., 2017; Salimans & Chen, 2018; Ecoffet et al., 2019; Florensa et al., 2017).

NLP with Human Feedback Learning from human feedback has been studied in the past in the context of bandit feedback (Nguyen et al., 2017; Sokolov et al., 2016), pairwise feedback (Scheurer et al., 2023; Chen et al., 2023) and other feedback forms (Kreutzer et al., 2018a; Sumers et al., 2021; Hancock et al., 2018; Wu et al., 2021). RLHF from has been an active area of research employing RL as the main strategy to align LMs with human preferences (Ouyang et al., 2022; Bai et al., 2022a; Bakker et al., 2022; OpenAI, 2023; Nakano et al., 2021; Wu et al., 2021; Stiennon et al., 2020; Ziegler et al., 2019). A remarkable result in this line of work is ChatGPT (OpenAI, 2023). The general process involves learning a preference reward model induced by human preferences and then finetuning with RL using this learned preference model.

LLM Finetuning from AI Feedback: Despite being easier to collect than expert data, high-quality human preference data collection is a key bottleneck of scaling RL finetuning for LLMs. A growing body of work enlists the help of LLMs to augment various parts of the RLHF procedure. ConstitutionalAI and RLAIF (Bai et al., 2022b; Lee et al., 2023a) explores using LLMs to generate preference datasets to do reward model training on while (Roit et al., 2023; Yang et al., 2023; Kwon et al., 2023) finds directly generating reward signals from another LLM to be effective. Separate from this literature, we investigate utilizing direct LLM feedback during the generation process, reminiscent of RL algorithms utilizing expert interactive feedback.

RL for Text Understanding and Generation: RL has been used to train text generation models for dialogue (Li et al., 2016), text simplification (Zhang & Lapata, 2017), machine translation (Kiegeland & Kreutzer, 2021; Wu et al., 2016; Shen et al., 2015), image captioning (Ren et al., 2017), question generation (Pang & He, 2021). RL has also been used to create models that take actions given a text such as for instruction following (Hermann et al., 2017; Misra et al., 2017), text games (Narasimhan et al., 2015; Côté et al., 2019; Ammanabrolu & Riedl, 2018), and code generation (Zhong et al., 2017). These methods typically use policy gradient based RL. Recently, (Ramamurthy et al., 2022) studied online RL for text generation across a wide range of tasks, specifically studying Proximal Policy Optimization (PPO) (Schulman et al., 2017). Although the results comparing RL and SL are mixed, we build upon their work and show the benefit of RL and ultimately RLGF outperforming SL and RL. Separately, (Snell et al., 2022) studies offline RL in the context of text generation whereas our work studies the online case.

B Additional Algorithms

A detailed algorithm for `LOLS` showing how to combine reinforcement learning and imitation learning differently than `D`^{2}`LOLS`. Rather than setting α to be the stopping time to switch from `AggreVaTeD` to `PPO`$^{++}$, we have a mixing probability of combining `AggreVaTeD` and `PPO`$^{++}$ at every iteration, $\alpha A^{\pi_\theta^t} + (1-\alpha)A^{\pi^g}(s,a)$. As discussed in Section 5, we find that `LOLS` underperforms `D`^{2}`LOLS`, even in practice.

Algorithm 4 `LOLS`: combine `PPO` and `AggreVaTeD`

```
1: Input: $\pi_\theta$, reference $\pi^g$, iterations T, dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$
2: Input: mixing parameter $\beta_1 \in [0,1]$, mixing parameter $\beta_2 \in [0,1]$, mixing prob $\alpha$
3: for t = 0,1,...,T-1 do
4:
   ▷ PPO$^{++}$
5:     Rollin with $\beta_1 \pi^g + (1-\beta_1)\pi_\theta^t$ starting from $x \sim \mathcal{D}$
6:     Rollout with $\pi_\theta^t$ to collect trajectories
7:     Update $V_\phi^{\pi_\theta^t}$ with trajectories and compute advantage estimates $A^{\pi_\theta^t}$
8:
   ▷ AggreVaTeD
9:     Rollin with $\beta_2 \pi_\theta^t + (1-\beta_2)\pi^g$ starting from $x \sim \mathcal{D}$
10:    Rollout with $\pi^g$ to collect trajectories
11:    Update $V_\phi^{\pi^g}$ with trajectories and compute advantage estimates $A^{\pi^g}(s,a)$
12:
   ▷ Mix Update
13:    Update $\pi_\theta$ using PPO loss with $\alpha A^{\pi_\theta^t} + (1-\alpha)A^{\pi^g}(s,a)$
```

C Additional Experimental Details

C.1 KL Reward Constraint

In addition to sequence-level task rewards, per-token KL rewards are applied to prevent the policy π from deviating too far from the pre-trained LM π_0, following the works Ziegler et al. (2019); Ouyang et al. (2022); Ramamurthy et al. (2022). Formally, regularized reward function is defined as: $\hat{R}(s_t, a_t, y) = R(s_t, a_t, y) - \lambda \mathrm{KL}\left(\pi(a_t|s_t)||\pi_0(a_t|s_t)\right)$ where $\mathrm{KL}\left(\pi(a_t|s_t)||\pi_0(a_t|s_t)\right) = (\log \pi(a_t|s_t) - \log \pi_0(a_t|s_t))$ and λ is the KL coefficient Ouyang et al. (2022). Note we used use a fixed KL coefficient rather than an adaptive controller.

C.2 Task Details

Task	Train/Val/Test	Prompt	Gen. Length
`IMDB`	25K/5K/5K	Partial movie review up to 64 tokens	48
`CommonGen`	32651/993/1497	"Generate a sentence with: " set of 3-5 concepts	20
`TL;DR`	117000/6450/6550	"TL;DR: "	50
`TL;DR Preference`	92500/3300/8300	"TL;DR: "	N/A

Table 3: Train, val, test splits, prompts, and max generation length used for each task.

IMDB: We experiment on the IMDB dataset for positive movie review generation. As shown in Table 3, the dataset consists of 25k training, 5k validation and 5k test prompts of movie review text with either positive or negative sentiment labels. As in put to our models, we use partial movie reviews that are at most 64 tokens long and ask the model to complete the review with a positive sentiment with at most 48 generated tokens.

CommonGen: CommonGen Lin et al. (2020b) is a common sense text generation task where the model is given a set of concepts (i.e. hockey, rink, game) and is asked to generate a semantically correct sentence using those concepts (i.e. the hockey team played a game at the rink). We follow the same splits as the dataset creators and refer the readers to Table 1 of Lin et al. (2020b) for more in-depth statistics of the dataset. In our experiments, we prompted out models with "*generate a sentence with:* " and generated at most 20 tokens. We chose this generation length based on the maximum token length of the references in the training dataset.

TL;DR Summarization: Following Stiennon et al. (2020), we evaluate on the summarization task. We use `CarperAI/openai_summarize_comparisons` for the preference reward training dataset and `CarperAI/openai_summarize_tldr` for the RL training dataset. For the SFT model that we use for our starting policy and our guide policy, we use the publicly available checkpoint `CarperAI/openai_summarize_tldr_sft`. We truncated/padded each prompt to 500 tokens on the GPT-J 6B tokenizer.

We first train our reward model using LoRA adapters. Our reward training is 1 epoch and where we got 70% accuracy on the test set. With this reward model we run all of our experiments where our policy and critic are both LoRA adapters trained on top of SFT checkpoint.

Win Rate: We calculated the win rate against the dataset references using Llama2-13B-chat (Touvron et al., 2023) publically available on HuggingFace. Following DPO (Rafailov et al., 2023), we prompt the model with instructions, 2 summaries (A) and (B), and instructions on how to answer. We randomize which summary is (A) or (B) when calculating the win rate over the test set. Below is our prompt skeleton:

```
<<SYS>>
You are an expert summary evaluator and can consistently
distinguish between good and bad summaries. You provide
informative, correct evaluations.
<<\SYS>>

Task: Judge the quality of two TLDRs, choose the options
among (A) or (B)
context: [context]
tldr (A): [summary 1]
tldr (B): [summary 2]
FIRST provide a one-sentence comparison of the two summaries,
explaining which you prefer and why. SECOND, on a new line,
state only (A) or (B) to indicate your choice. Your
response should use hte format:
Comparison: <one-sentence comparison and explanation>
Preferred: <(A) or (B)>
```

C.3 IMDB - Algorithm Details

Table 4 lists the hyperparameters used in our IMDB experiments. Note that we used the same parameters here for all guide policies. Across all algorithms, we shared the same parameters as the ones we used for our `PPO` baseline. Finally, we use top-k sampling with $K = 50$ as the decoding method and for fair comparison, we keep this setting for all methods.

Setting	Values
model	GPT2
`PPO`	steps per update: 1280 total number of steps: 128000 batch size: 64 epochs per update: 5 learning rate: 1e-6 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 λ: 0.001 η: 0.1
`PPO`$^{++}$	Mixing Parameter (β): 0.2
`AggreVaTeD`	Mixing Parameter (β): 0.8
`LOLS`	Mixing Probability (α): 0.8
`D`^{2}`LOLS`	Stopping Time Iteration (α): 20
decoding	sampling: true top k: 50 min length: 48 max new tokens: 48
tokenizer	padding side: left truncation side: left max length: 64

Table 4: Hyperparameters used for IMDB. Note that `PPO`$^{++}$, `AggreVaTeD`, `LOLS`, and `D`^{2}`LOLS` all share the same PPO parameters. All processes use the same decoding and tokenizer parameters.

C.4 CommonGen - Algorithm Hyperparameters

Setting	Values
model	T5
`PPO`	steps per update: 663,552 total number of steps: 66,355,200 batch size: 2048 epochs per update: 1 learning rate: Linear decay 1e-5 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.4 value function coeff: 3.0 λ: 0.001 η: 0.1
`PPO`$^{++}$	Mixing Parameter (β): 0.2
`AggreVaTeD`	Mixing Parameter (β): 0.8
`LOLS`	Mixing Probability (α): 0.8
`D`^{2}`LOLS`	Stopping Time Iteration (α): 20
decoding	num beams: 5 min length: 5 max new tokens: 20
tokenizer	padding side: left max length: 20

Table 5: Hyperparameters used for CommonGen. Note that `PPO`$^{++}$, `AggreVaTeD`, `LOLS`, and `D`^{2}`LOLS` all share the same PPO parameters. All processes use the same decoding and tokenizer parameters.

Table 5 lists the hyperparameters used in our CommonGen experiments. Note that we used the same parameters here for all guide policies. Across all algorithms, we shared the same parameters as the ones we used for our `PPO` baseline. Finally, we use beam search with the number of beams = 5 as the decoding method for inference. Note that for training, we still used softmax sampling with default temperature. For fair comparison, we keep this setting for all methods. Finally, note that for CommonGen, we set the KL coefficient to 0.

C.5 TL;DR Summarization - Algorithm Hyperparameters

Table 6 lists the hyperparameters used in our TL;DR summarization experiments. Note that we used the same parameters here for all guide policies. Across all algorithms, we shared the same parameters as the ones we used for our `PPO` baseline. The reward LoRA adapter was used to pretrain our model. During RL training, we kept the reward adapter frozen.

Setting	Values
model	GPT-J
PPO	steps per update: 70,400 total number of steps: 7,040,000 batch size: 128 epochs per update: 4 learning rate: 1e-5 discount factor: 1.0 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.2 λ: 0.002 η: 0.1
PPO^{++}	Mixing Parameter (β): 0.6
AggreVaTeD	Mixing Parameter (β): 1.0
decoding	max new tokens: 50
tokenizer	padding side: left truncation side: right max length: 50
RM LoRA	r: 32 α: 32 dropout: 0.0
Policy LoRA	r: 8 α: 64 dropout: 0.1
Critic LoRA	r: 8 α: 64 dropout: 0.1

Table 6: Hyperparameters used for TL;DR summarization. Note that PPO^{++}, AggreVaTeD all share the same PPO parameters.

D IMDB Qualitative Examples

Sample 1
Prompt: not a complex character, but totally rotten. The relation between him and Gisburne is just hilarious. Actually, just looking at de Rainault sitting in his throne, bored, glaring, makes me laugh even before he has said anything. Another actor that deserves extra praise is Ray Winstone as Will Scarlet. You
GPT2: know, when you think he’s a bad actor, and when you think the script is all about the bad actors. Scarlet is the typical actor with a weak, pathetic voice, who fails to show emotion or self-respect, and who
BC: will not be disappointed. The opening scenes are amazing and the final scene of the film is amazing. Every single minute of this film is a treat with every emotion the characters are experiencing. From the first frame of the
PPO: know he likes old movies. I gave this film a 3 and a half out of 10. And, of course, I don’t blame you for falling asleep! I gave it a 10 because of my favorite
PPO+: see he seems only interested in money, which really makes the two seem like friends. One particularly funny scene involves Winstone and the rest of the cast in their studio for the first time since "Rampage". I absolutely LOVE the way
AggreVaTeD: will laugh heartily; he is brilliant and always entertaining. I also think the film is just wonderful. I love what Van Dyke is able to bring to the film. If you want to go great, I’m sure Van Dyke
LOLS: great film. I absolutely loved his style and direction. One of my favorite movies of all time. The first 25 minutes are really good and really enjoyable. It’s actually very touching and engaging. The cast, including James Gandolfini
D2LOLS: could never have come across this again. It was one of the funniest films I’ve ever seen, and it remains my favorite movie ever. I am absolutely honored and very excited to see it again!!! I really love it and look forward

Sample 2
Prompt: comes from Zekeria Ebrahimi, who plays the young Amir. It’s not that it’s bad, it’s just that it’s not as effective as it could be. He just can’t translate the guilt that consumes Amir to the audience. In fact, I think I might have been a little lost at
GPT2: first. The movie has too much going for it, too many unanswered questions and some of the characters not getting the emotional payoff in a reasonable enough manner. It’s nice to see a filmmaker who isn’t a hater, but I feel
BC: the second half as it wasn’t shown at all until the very end, at best. I liked the plot (which I think was very well done, considering it’s a "bad cop story"). It’s not
PPO: first. After listening to the great Ghazala, who’s brilliant in her acting, I am very impressed with her acting performance. I loved how you have a young and very attractive girl named Zekeria who is portrayed quite well and
PPO+: last week’s deadline. It’s quite a clever and good film, as well as an entertaining one. 4/10. Recommended. 8/10
AggreVaTeD: first. The movie really deserves the credit - it really is a very moving story and a really moving movie that is an amazing film. I personally love Amir and think it’s an extremely powerful film. It’s one of the things that makes
LOLS: some points. As a whole it’s pretty enjoyable and is very well-acted and well edited, and is quite a compelling, quirky and fun film. And just as important, I found some very well-acted, and well-acted
D2LOLS: first, when Amir realizes what’s really going on. There are many terrific performances in this beautifully directed film. I can’t say which is one of the best, which is certainly a shame because it’s an enjoyable

Sample 3
Prompt: Well, now I finally HAVE seen the true garbage version - and it’s Paul Schrader’s DOMINION: PREQUEL TO THE EXORCIST. It was relentlessly talky, uninteresting, and insipid. Stellan Skarsgard’s troubled priest was nowhere near as
GPT2: interesting as the other characters in his book - and most of the scenes in the book were just so poorly edited and over-the-top; the acting was too, horribly bad, and most of the dialogue was thrown out in a couple
BC: well developed as the novel and story had - but did bring some of the best bits to the screen. I watched the movie expecting a story - "The God" - but found it just an excuse for schlock
PPO: good (I guess), if anything, the sort of low-brow, uninteresting sort of horror that made an original film "The Cat With the Feeling" look like some sort of cheesy 80’s horror flick from 1999.
PPO+: exciting. And in addition, the only really really interesting character in this movie was Paul Schrader, who is at his worst as a creepy, self-absorbed man in a bar. I can’t say I’ve ever liked a
AggreVaTeD: original or entertaining as I’d anticipated, and in the end, I got to see some interesting and engaging dialog. I loved that scene with Schrader explaining how it would be fun for him to make a movie that would make people think twice
LOLS: good as his master. Still, the film’s message is very much worth exploring. It was the perfect love story for me, as you would expect. I’m an evangelical and am an excellent fan of all things Christian. I love reading
D2LOLS: brilliant as the others, as well as beautifully acted. Of course, some points in the film are so well acted and acted that, despite all its flaws, this is a very touching film. The cinematography is

Figure 5: **Qualitiative Analysis of IMDB**

E CommonGen Qualitative Examples

Sample 1
Prompt: generate a sentence with: dress perform stage.
T5: dress up and perform on stage
BC: A young girl in a black dress is performing on a stage.
PPO: performers dress up on the stage.
PPO+: a woman performing on the stage dressed up.
AggreVaTeD: young woman performing on stage in a black dress.
LOLS: people performing on the stage in a dressed up.
D2LOLS: A woman is performing on the stage in a dress.

Sample 2
Prompt: generate a sentence with: ball fall throw.
T5: ball falls off the ground
BC: A man throws a ball that has fallen to the ground.
PPO: the ball falling from the throw.
PPO+: the ball is thrown and the ball falls.
AggreVaTeD: A man is throwing the ball into the ground.
LOLS: throwing the ball after it fell from the sky.
D2LOLS: A man is throwing the ball after it fell.

Sample 3
Prompt: generate a sentence with: arm chest fold.
T5: arm folds in the chest
BC: He folds his arms over his chest, then he folds his arms over.
PPO: folded the arms in the chest.
PPO+: a man with his arms folded in the chest.
AggreVaTeD: folding his arm over his chest.
LOLS: A man folds his arms in the chest.
D2LOLS: A man with his arms folded in the chest.

Figure 6: **Qualitiative Analysis of CommonGen**

F ADDITIONAL RESULTS

We further go on to compare starting from either a SFT warmstarted policy of AggreVaTeD. We show that for IMDB, we obtain better performance when performing RL after AggreVaTeD rather than SFT.

Alg	Semantic Score
SFT+PPO	0.767 ± 0.018
AggreVaTeD + PPO	0.863 ± 0.007
SFT+PPO^{++}	0.883 ± 0.011
D^{2}LOLS(i.e. AggreVaTeD + PPO†)	**0.896 ± 0.012**

Table 7: Warmstarting with SFT or AggreVaTeD: Results of running PPO or PPO^{++} after warmstarting with either SFT or AggreVaTeD. For both PPO and PPO^{++}, warmstarting with AggreVaTeD yields the better results.