Learning to Generate Better Than Your LLM

Jonathan D. Chang*
Department of Computer Science
Cornell University
jdc396@cornell.edu

Kiante Brantley*
Department of Computer Science
Cornell University
kdb82@cornell.edu

Rajkumar Ramamurthy
Fraunhofer IAIS
rajkumar.ramamurthy@iais.fraunhofer.de

Dipendra Misra
Microsoft Research New York
dipendra.misra@microsoft.com

Wen Sun
Department of Computer Science
Cornell University
ws455@cornell.edu

Abstract

Reinforcement learning (RL) has emerged as a powerful paradigm for fine-tuning Large Language Models (LLMs) for conditional text generation. In particular, recent LLMs such as ChatGPT and GPT-4 can engage in fluent conversations with users by incorporating RL and feedback from humans. Inspired by *learning-to-search* algorithms and capitalizing on key properties of text generation, we seek to investigate reinforcement learning algorithms beyond general purpose algorithms such as Proximal policy optimization (PPO). In particular, we extend RL algorithms to allow them to interact with a dynamic black-box guide LLM such as GPT-3 and propose RL *with guided feedback* (RLGF), a suite of RL algorithms for LLM fine-tuning. We experiment on the IMDB positive review and CommonGen text generation task from the GRUE benchmark. We show that our RL algorithms achieve higher performance than supervised learning (SL) and default PPO baselines, demonstrating the benefit of interaction with the guide LLM. On CommonGen, we not only outperform our SL baselines but also improve beyond PPO across a variety of lexical and semantic metrics beyond the one we optimized for. Notably, on the IMDB dataset, we show that our GPT-2 based policy outperforms the zero-shot GPT-3 oracle, indicating that our algorithms can learn from a powerful, black-box GPT-3 oracle with a simpler, cheaper, and publicly available GPT-2 model while gaining performance.

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 (86), write articles from short descriptions (29), generate code (26), follow robot commands (34), solve puzzles (12), and even showcased as assistive models for education (37) and healthcare (42). These LLMs are typically trained in a task-agnostic way with supervised learning (SL) to maximize an auto-regressive language modeling objective. A combination of large models and datasets has enabled LLMs to generalize across tasks, generating incredibly fluent, human-like text.

*Equal contribution

Preprint. Under review.

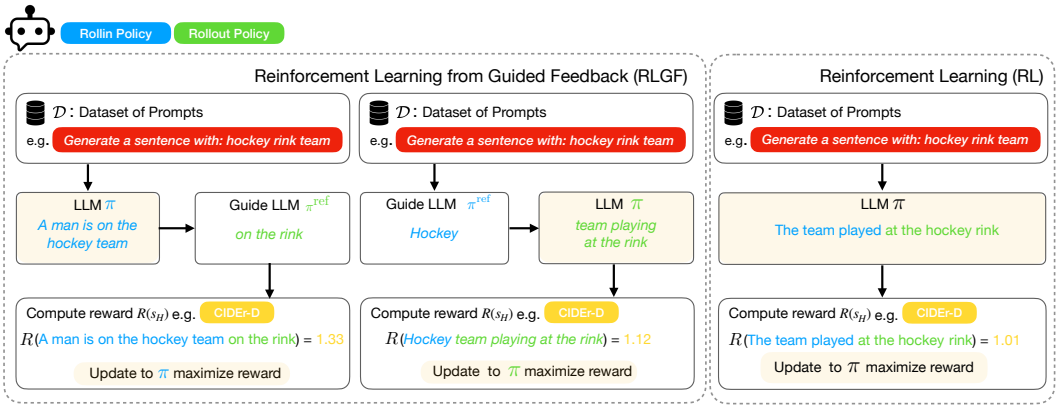

Figure 1: (Top) Reinforcement Learning with Guided Feedback (RLGF) flow chart showing how breaking up generations into two parts, rollins and rollouts done by different LLMs opens up a rich framework of interaction when compared to (Bottom) Reinforcement Learning (RL). RL can be instantiated by setting both the rollin and rollout policy to be the same language model.

However, using SL to train sequence prediction models presents a challenging mismatch between the training and testing regimes, namely, *metric mismatch* (81). 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. Indeed, OpenAI fine-tunes LLMs with RL on a learned metric from human feedback to align LLMs to human intentions, leading to the great success of ChatGPT (53). Recently, GRUE benchmark (59) systematically studied RL versus SL when finetuning LLMs on downstream tasks with pre-defined rewards. GRUE's preliminary results demonstrate the benefit of RL when finetuning LLMs, leading to the release of popular codebases such as RL4LMs (59) and TRLX (13), that enables RL for language models. However, ChatGPT, RL4LMs, TRLX use vanilla policy gradient methods which are known to be sample inefficient and sensitive to local minima due to the combinatorially large search space of natural language generation (59).

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 leverage 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 existing pre-trained 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. Figure 1 showcases RLGF's different ways of integrating LLM guidance and highlights the main differences between RLGF and RL.

We evaluate on two conditional text-generation tasks from the GRUE benchmark (59). The first is the IMDB task where the goal is to generate a positive and fluent review given an initial context. The second is the CommonGen task where the goal is to write a fluent text that uses a given set of words. For both, 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, showing RLGF learning stronger text generators by leveraging a guide policy during training.

In particular, one of our algorithms, Direct and Differentiable Locally Optimal Learning to Search (D^2LOLS), achieves the best performance overall, even **outperforming PPO on a variety of lexical and semantic metrics beyond the ones we optimized for** in CommonGen. Notably, when using GPT-3 as our guide policy, our **GPT-2 based policy outperforms GPT-3 policy** on the IMDB task. This opens up the potential for RLGF to leverage a general-purpose, black box model as a guide

policy to fine-tune a cheaper model that is freely available and get a performance that can match or even exceed the guide policy.

2 Related Work

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

Imitation Learning for Structured Prediction: Algorithms such as Schedule Sampling (SS) (10), methods using SS (23; 47; 28), SEARNN (41), Bridging the Gap (85), Mixer (60) been inspired by IL for structured prediction algorithms DAGGER (64), DAD (80), and SEARN (22). Our work is inspired by AggreVaTeD (77) (Differentiable AggreVaTe (63)) where the algorithm makes use of differentiable policies and multi-step feedback rather than immediate one-step predictions to imitate.

Reinforcement Learning for Text Understanding and Generation: RL has been used to train text generation models for dialogue (43), text simplification (87), machine translation (38; 83; 70), image captioning (62), question generation (55). RL has also been used to create models that take actions given a text such as for instruction following (33; 48), text games (51; 20; 4), and code generation (88). These methods typically use policy gradient based RL. Recently, (59) studied online RL for text generation across a wide range of tasks, specifically studying Proximal Policy Optimization (PPO) (69). 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, (72) studies offline RL in the context of text generation whereas our work studies the online case.

NLP with Human Feedback Learning from human feedback has been studied in the past in the context of bandit feedback (52; 73), pairwise feedback (67; 16) and other feedback forms (40; 40; 75; 32; 82). RLHF from has been an active area of research employing RL as the main strategy to align LMs with human preferences (54; 7; 8; 53; 50; 82; 74; 90). A remarkable result in this line of work is ChatGPT (53). The general process involves learning a preference reward model induced by human preferences and then finetuning with RL using this learned preference model.

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) (22; 21; 19; 61).

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 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 $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. 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 $A^{\pi^g}(s, a)$.

4 Reinforcement Learning from Guided Feedback

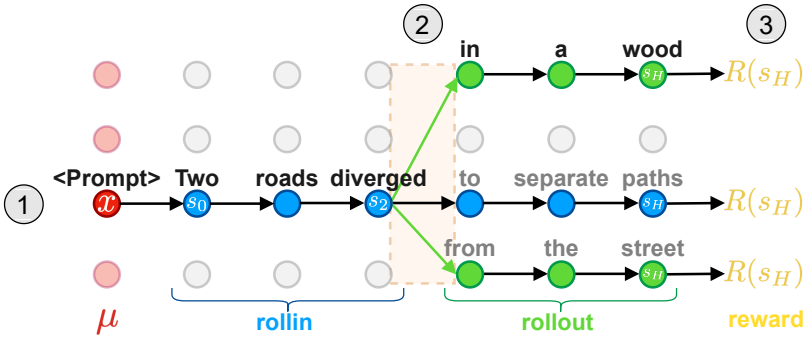

Figure 2: RLGF's main mechanism of incorporating guidance through interactions between two LLMs: rollin and rollout policies. (1) the rollin policy generates a trajectory. (2) the rollout policy restarts to a sampled point in the generation (i.e. s_2) and completes the generation. (3) the rollout policy 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 2. The rollin policy is used to generate sequences that the rollout policy evaluates. More 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. In particular, 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^{++}

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

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 (11): Conservative Policy Iteration (CPI) (35). 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 a modified 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 $\beta\pi^{ref} + (1-\beta)\pi_\theta$ that interpolates between the current policy and the guide policy, with some probability β, for rollin (see Algorithm 1 Line 3). This ensures that with probability β, PPO^{++} is executing the default PPO update, making sure PPO^{++} in practice never underperforms default PPO.

Algorithm 2 AggreVaTeD

1: **Input:** π_θ, guide π^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 $\beta\pi_\theta^t + (1-\beta)\pi^g$ starting from $x \sim \mathcal{D}$
4: Rollout with π^g to collect trajectories
5: Update $V_\phi^{\pi^g}$ with trajectories and compute advantage estimates A^{π^g}
6: Update π_θ using PPO loss with A^{π^g}
7: **return** π_θ

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 (77), a differentiable policy gradient version of AggreVaTe (Aggregate Values to Imitate (63)) as seen in Algorithm 2. AggreVaTeD is an API algorithm similar to CPI and also uses a mixture policy $\beta\pi^{ref} + (1-\beta)\pi_\theta$ for rollin. This algorithm first generates rollins with the mixture policy to collect sequences. Then AggreVaTeD uses rollouts with the guide policy to restart and evaluate the quality of the generated rollins. Rolling out with π^g ensures that the LLM policy π_θ can be *at least* as good as and potentially better than the guide policy π^g.

Algorithm 3 D^{2}LOLS

1: **Input:** π_θ, guide π^g, iterations T, dataset $\mathcal{D} = \left\{(x^i, y^i)\right\}_{i=1}^N$
2: Run $\pi_\theta^1 =$ AggreVaTeD$(\pi_\theta, \pi^g, \alpha T, \beta_1, \mathcal{D})$
3: Run $\pi_\theta^2 =$ PPO$^{++}(\pi_\theta^1, \pi^g, (1-\alpha)T, \beta_2, \mathcal{D})$
4: **return** π_θ^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) 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^{++} for the remainder of the algorithm, fixing LOLS's issue arising from interweaving guidance.

5 Theoretical Justification

In this section, we analyze the theoretical properties of the 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 for each of our schemes are derived from the original algorithms that each method has built upon.

Interactive Imitation Learning: AggreVaTeD In our interactive imitation learning 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 (77) to incorporate a PPO policy gradient loss. The overall idea of AggreVaTeD 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 we our LLM 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 (e.g., $\max_a A^{\pi^g}(s,a)$ 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 (77) and is also confirmed in our experiments. We show that even when fine-tuning GPT2 with GPT3 as the guide policy, i.e. $\pi^g\notin\Pi$, an AggreVaTeD GPT2 model learns to outperform GPT3.

Reinforcement Learning with 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 globally 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^{++} as seen in 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 (35; 6; 3; 1) ensures that as long as the globally optimal policy is covered by the restart distribution, we can learn to perform as well as the globally 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 (35). 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). Our experiments verify that restarting based on states from $d^{\pi^g}_{\mathcal{D}}$ improves the performance.

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) [2] (68) or CPI (35), then our algorithm has a guaranteed monotonic policy improvement. This means that upon convergence, we achieve the following two properties: (1) our learned policy is at least as good as the guide policy π^g (and can be better due to the non-negative margin), (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 (as otherwise TRPO and CPI can keep making improvement).

[2] 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 (35) to ensure monotonic improvement.

Algorithms	IMDB Sentiment *Semantic and Fluency Metrics*		CommonGen *Lexical and Semantic Metrics*				
	Sentiment Score ↑	Perplexity ↓	Bleu-4	BERTScore	CIDEr-D	SPICE	**SPIDER**
Zero-Shot	0.484 ± 0.002	32.545 ± 0.001	0.160	0.929	1.101	0.257	1.358
BC	0.552 ± 0.001	35.668 ± 0.001	0.218	0.945	1.433	0.309	1.752
PPO	0.737 ± 0.014	34.114 ± 0.225	0.236	0.938	1.572	0.300	1.872
PPO^{++}	0.887 ± 0.015	37.727 ± 0.502	0.243	0.936	1.577	0.296	1.873
AggreVaTeD	0.788 ± 0.023	35.199 ± 0.176	0.244	0.945	1.521	0.304	1.825
LOLS	0.887 ± 0.019	36.846 ± 0.450	0.243	0.937	1.555	0.297	1.852
D^{2}LOLS	$\mathbf{0.896 \pm 0.012}$	37.256 ± 0.230	**0.254**	**0.947**	**1.644**	**0.315**	**1.959**
BC+PPO	0.767 ± 0.018	39.032 ± 0.269	0.261	0.949	1.652	0.322	1.974
BC+PPO^{++}	0.883 ± 0.011	41.461 ± 1.391	0.274	**0.951**	1.680	0.324	2.004
BC+AggreVaTeD	0.852 ± 0.019	39.118 ± 0.531	0.268	**0.951**	1.650	0.321	1.971
BC+LOLS	0.898 ± 0.023	41.701 ± 1.202	0.260	**0.951**	1.658	0.322	1.980
BC+D^{2}LOLS	$\mathbf{0.903 \pm 0.015}$	40.592 ± 1.502	**0.270**	**0.951**	**1.692**	**0.328**	**2.020**

Table 1: **IMDB and CommonGen Results:** We compute the mean and standard deviation over 5 seeds for the IMDB task and compute 1 seed for the CommonGen task. For our reward function each task we use the bold metric. The zero-shot model is the performance of the pretrained model used for IMDB and CommonGen, GPT-2 and T5 respectively. BC+Alg indicates running Alg after supervised finetuning. BC is used as our guide policy π^g for all experiments.

There exist several algorithms in the literature that combine RL and IL (17; 76; 14; 58; 49). D^{2}LOLS (Algorithm 3) is a modification of LOLS (Algorithm 4) to incorporate a policy gradient loss. 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 best of RL and IL, 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. Note that the simplest approach D^{2}LOLS works the best, and achieves the guarantee of LOLS without the additional assumption of the above gap being small.

6 Experiments

The research questions that we aim to answer in our experimentation section are: **(1)** how does RLGF algorithms perform against the baselines as well as amongst each other; **(2)** how extensively do RLGF algorithms rely on interaction with a guide policy? **(3)** qualitatively, how do algorithms utilizing a guide policy perform when faced with prompts of varying difficulty levels, such as easy and hard; and **(4)** is RLGF able to leverage powerful black box oracles like GPT-3. To answer these questions, we conduct experiments on two tasks proposed in the GRUE (General Reinforced-Language Understanding Eval) benchmark (59): IMDB - text continuation and CommonGEN - generative commonsense. We investigate the performance of RLGF algorithms against two baseline algorithms: behavior cloning (BC) and PPO (with KL constraint (90)). Behavior cloning is the simplest imitation learning algorithm that performs supervised learning updates using the task labeled dataset. We use the baseline BC policy as our guide policy π^g.

6.1 Task Setup

The objective of the IMDB task is to generate fluent and positively sentiment-ed text continuations for IMDB (46) movie reviews prompts. We use a sentiment classifier (66) 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 BC baselines, we consider only the examples with positive labels. We chose GPT2 as the base language model (LM)

Algorithms	IMDB Sentiment *Semantic and Fluency Metrics*					
	Sentiment Score ↑	Perplexity ↓	Sentiment Score ↑	Perplexity ↓	Sentiment Score ↑	Perplexity ↓
Mixture Policy	$\beta = 0.2$		$\beta = 0.5$		$\beta = 0.8$	
AggreVaTeD	0.718 ± 0.823	37.055 ± 0.379	0.742 ± 0.233	36.787 ± 0.230	$\mathbf{0.788 \pm 0.023}$	35.199 ± 0.176
PPO^{++}	$\mathbf{0.887 \pm 0.015}$	37.727 ± 0.502	0.854 ± 0.012	37.023 ± 0.044	0.821 ± 0.037	37.506 ± 0.767
Mixing Time	$\alpha = 20\%$:(20/80) iter		$\alpha = 50\%$:(50/50) iter		$\alpha = 80\%$:(80/20) iter	
D^{2}LOLS	$\mathbf{0.896 \pm 0.012}$	37.256 ± 0.230	0.886 ± 0.006	37.521 ± 0.547	0.803 ± 0.070	36.482 ± 1.545

Table 2: **IMDB Mixture Policy Analysis**: The top row shows the mean and standard deviations over 3 random seeds for testing the sensitivity of the the mixture policy parameter for both PPO^{++} and AggreVaTeD . **IMDB Mixing Time**: The bottom row shows the mean and standard deviations over 3 random seeds for testing D^{2}LOLS mixing time parameter. For example, 20/80 would be 20 iterations of AggreVaTeD and then 80 iterations of PPO /PPO^{++} .

for this task as it is more suited for text continuation than encoder-decoder models. We evaluate all algorithms on two metrics: sentiment reward score and fluency metric (perplexity).

CommonGen (44) is a challenging constrained text generation task that tests the ability of generative common sense reasoning. For example, given an input set of concepts (dog, frisbee, catch, throw), the task is to generate coherent natural sentences like *"A dog jumps to catch a thrown frisbee"*. We optimize the SPIDER (45) reward function which is a weighted combination of the CIDEr-D metric (79) and SPICE (5). We chose T5-base as the base LLM since it is well-suited for structure-to-text tasks. We additionally note that concept set inputs are prefixed with "generate a sentence with:". The T5 model tends to reward hack (59; 71) – repeating the original prompt and concept – when optimizing the task reward. To address this issue, we warm-started all algorithms with a BC model using 10% of the expert demonstrations, to provide the model with task-specific data. We evaluate all algorithms on four metrics: BLEU (56), CIDEr-D (79), SPICE (5), and BERTScore (84).

6.2 Experimental Results

The main results are shown in Table 1 for both tasks across all algorithms. First, we find that all RL based algorithms outperform BC across all lexical and semantic metrics for both tasks. For both tasks, our π^g (is set to our baseline BC) is sub-optimal, performing worse than our RL baseline across most metrics. Despite this sub-optimality of π^g, for IMDB all RLGF algorithms outperform PPO and for CommonGen, D^{2}LOLS outperforms PPO . Algorithms that combine RL and IL, LOLS and D^{2}LOLS, outperform other RL algorithms on the reward function being optimized. We also observe this performance trend when finetuning after a supervised pre-training step. Furthermore, we see that the different mixing strategies discussed Section 5 has an effect on performance.

As our theory suggests, AggreVaTeD is able to improve beyond our guide policy, providing an alternative as a warm-starting methodology to warm-starting with BC. As shown by Table 7, we see that warm-starting with AggreVaTeD leads to higher performance on IMDB than warm-starting with BC, a popular learning strategy when performing RL for language (59; 74; 54). 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. Finally, the combination of these two, D^{2}LOLS, achieves the best of both worlds and fully leverages the full capabilities of utilizing a guide policy.

Balancing Interaction Budget and Learning An essential aspect of our algorithms lies in their interaction with a guide policy, leading us to examine the extent of interaction required for the algorithms to attain a policy that is comparable to the guide policy. For example, when considering the IMDB task, we observe that PPO^{++}only necessitates a minimal 20% interaction, while AggreVaTeD, as an imitation learning approach, demands a significantly higher level of interaction at 80%. With this insight, we explored further how to allocate the iteration budget for D^{2}LOLS. The results presented in the Table 2 highlight the criticality of initially performing warm-start with AggreVaTeD, utilizing 20% of the iterations, before proceeding to fine-tune with PPO^{++}.

Impact of Prompt Difficulty In order to answer question **(3)**, we conducted a study to assess their performance across varied difficulty levels of input prompts. Our evaluation was carried out on

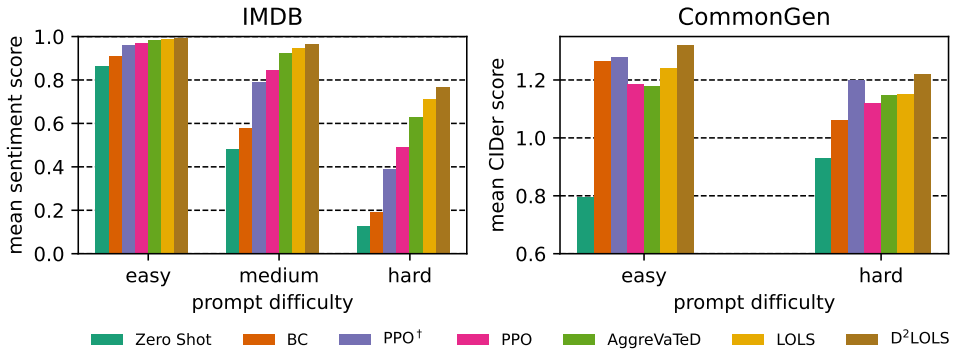

Figure 3: Comparison of sentiment and CIDer scores for different algorithms grouped by prompt difficulty on IMDB and CommonGen tasks respectively. The performance gap between easy and hard prompts is evident for BC, and PPO^{++}, while our proposed algorithms AggreVaTeD, LOLS and D^{2}LOLS exhibit a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

both the IMDB dataset and CommonGen where we categorized the prompts based on their difficulty level. For IMDB, the difficulty of any prompt is determined by analyzing its sentiment score, with higher scores indicating easier prompts for the model in generating continuations with positive sentiment. For this, we classified the prompts into three difficulty levels: *easy*, *medium*, *hard* and computed the average sentiment metric for each difficulty level. Similarly, for CommonGen, we classify the prompts into *easy* and *hard* based on the number of unseen concepts in the prompt. Figure 3 presents a comparison of scores for different algorithms grouped by prompt difficulty on IMDB and CommonGen tasks. The results reveal a notable performance gap between easy and hard prompts for algorithms such as BC and PPO, 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.

	IMDB Sentiment					
Alg	Zero-Shot GPT2	Zero-Shot GPT3	PPO^{++}	AggreVaTeD	LOLS	D^{2}LOLS
Score	0.484	0.77	0.867	0.761	0.867	**0.879**

Table 3: **GPT3 Guide Policy:** Sentiment scores on IMDB when finetuning a GPT2 model with a GPT3 guide policy for one seed. RLGF algorithms are able to match and even outperform GPT3.

GPT Models as Guide Policies To answer question **(4)**, we use a black-box GPT3 model as our guide policy. In Table 3, we see all RLGF algorithms fine-tuning a GPT2 model to be as good or better than GPT3. This suggests a potential impact of RLGF to be the use of general purpose, black-box guide policies to train a smaller, cheaper model to be very adept at a specific task.

7 Discussion and Future Work

In this work, we presented a unifying framework of incorporating a guide policy to enhance reinforcement learning for natural language generation. We experimentally verified that modern LLMs are proficient enough at providing guided feedback to improve learning. This opens up incredible possibilities to scale the amount of interactive feedback that was prohibitively expensive to get before. One notable outcome of this framework is the potential to use black-box feedback from powerful general-purpose LLMs to train a very competitive smaller and cheaper model. Even though we demonstrate on specific downstream tasks for this work, our algorithms in principle can be applied for general-purpose text reasoning, but it remains a challenging open question about what a good guide policy for this setting may be. 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.

Broader Impact We focus on LLM fine-tuning. No human user is involved during our training, and we present no human study in this work. During testing, we evaluate the model on these benchmarks. Our methods doesn't explicitly address common issues such as hallucination or harmful societal bias, which are common with LLMs. These are all relevant topics of work; however, their scope is orthogonal to our study. We do believe that standard LLM safeguards can also be applied to any LLM trained with our method since the learned LLM has the interface as any other LLM. We advocate caution and testing to control for various societal effects if deployed.

8 Acknowledgements

We would like to acknowledge the support of NSF under grant IIS-2154711. Kiante Brantley is supported by NSF under grant No. 2127309 to the Computing Research Association for the CIFellows Project. Rajkumar Ramamurthy is funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence.

References

[1] A. Agarwal, N. Jiang, and S. M. Kakade. Reinforcement learning: Theory and algorithms. 2019.

[2] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In *Conference on Learning Theory*, pages 64–66. PMLR, 2020.

[3] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift. *Journal of Machine Learning Research*, 22(1):4431–4506, 2021.

[4] P. Ammanabrolu and M. O. Riedl. Playing text-adventure games with graph-based deep reinforcement learning. *arXiv preprint arXiv:1812.01628*, 2018.

[5] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic Propositional Image Caption Evaluation. In *Proceedings of European Conference on Computer Vision*, 2016.

[6] J. Bagnell, S. M. Kakade, J. Schneider, and A. Ng. Policy Search by Dynamic Programming. *In Proceedings of Neural Information Processing Systems*, 2003.

[7] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[8] M. Bakker, M. Chadwick, H. Sheahan, M. Tessler, L. Campbell-Gillingham, J. Balaguer, N. McAleese, A. Glaese, J. Aslanides, M. Botvinick, et al. Fine-tuning Language Models to Find Agreement among Humans with Diverse Preferences. *Proceedings of Neural Information Processing Systems*, 2022.

[9] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[10] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *In Proceedings of Neural Information Processing Systems*, 2015.

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

[12] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of Artificial General Intelligence: Early Experiments with Gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[13] CarperAI. https://github.com/carperai/trlx, 2023.

[14] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, and J. Langford. Learning to Search Better than your Teacher. In *Proceedings of International Conference on Machine Learning*, 2015.

[15] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III, and J. Langford. Learning to Search Better than your Teacher. In *Proceedings of International Conference on Machine Learning*, pages 2058–2066, 2015.

[16] A. Chen, J. Scheurer, T. Korbak, J. A. Campos, J. S. Chan, S. R. Bowman, K. Cho, and E. Perez. Improving code generation by training with natural language feedback. *arXiv preprint arXiv:2303.16749*, 2023.

[17] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots. Fast Policy Learning through Imitation and Reinforcement. *arXiv preprint arXiv:1805.10413*, 2018.

[18] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[19] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 111–118, Barcelona, Spain, July 2004.

[20] M.-A. Côté, A. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, M. Hausknecht, L. El Asri, M. 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*, pages 41–75. Springer, 2019.

[21] H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction as classification. In *NIPS Workshop on Advances in Structured Learning for Text and Speech Processing, Whistler, Canada*, 2005/// 2005.

[22] H. Daumé, J. Langford, and D. Marcu. Search-based Structured Prediction. *Machine learning*, 75:297–325, 2009.

[23] D. Duckworth, A. Neelakantan, B. Goodrich, L. Kaiser, and S. Bengio. Parallel Scheduled Sampling. *arXiv preprint arXiv:1906.04331*, 2019.

[24] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[25] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

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

[27] D. Go, T. Korbak, G. Kruszewski, J. Rozen, N. Ryu, and M. Dymetman. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215*, 2023.

[28] K. Goyal, C. Dyer, and T. Berg-Kirkpatrick. Differentiable Scheduled Sampling for Credit Assignment. *arXiv preprint arXiv:1704.06970*, 2017.

[29] T. Goyal, J. J. Li, and G. Durrett. News Summarization and Evaluation in the Era of Gpt-3. *arXiv preprint arXiv:2209.12356*, 2022.

[30] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[32] B. Hancock, M. Bringmann, P. Varma, P. Liang, S. Wang, and C. Ré. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, page 1884. NIH Public Access, 2018.

[33] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. M. Czarnecki, M. Jaderberg, D. Teplyashin, et al. Grounded Language Learning in a Simulated 3D World. *arXiv preprint arXiv:1706.06551*, 2017.

[34] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proceedings of Annual Conference on Robot Learning*, 2022.

[35] S. Kakade and J. Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of International Conference on Machine Learning*, 2002.

[36] M. Khalifa, H. Elsahar, and M. Dymetman. A distributional approach to controlled text generation. *arXiv preprint arXiv:2012.11635*, 2020.

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

[38] S. Kiegeland and J. Kreutzer. Revisiting the weaknesses of reinforcement learning for neural machine translation. *arXiv preprint arXiv:2106.08942*, 2021.

[39] T. Korbak, H. Elsahar, G. Kruszewski, and M. Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *arXiv preprint arXiv:2206.00761*, 2022.

[40] J. Kreutzer, S. Khadivi, E. Matusov, and S. Riezler. Can neural machine translation be improved with user feedback? *arXiv preprint arXiv:1804.05958*, 2018.

[41] R. Leblond, J.-B. Alayrac, A. Osokin, and S. Lacoste-Julien. SEARNN: Training RNNs with Global-local losses. *arXiv preprint arXiv:1706.04499*, 2017.

[42] P. Lee, S. Bubeck, and J. Petro. Benefits, Limits, and Risks of GPT-4 as an AI Chatbot for Medicine. *New England Journal of Medicine*, 388(13):1233–1239, 2023.

[43] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2016.

[44] B. Y. Lin, W. Zhou, M. Shen, P. Zhou, C. Bhagavatula, Y. Choi, and X. Ren. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In *Findings of Association for Computational Linguistics: EMNLP*, 2020.

[45] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy. Improved Image Captioning via Policy Gradient Optimization of Spider. In *Proceedings of International Conference on Computer Vision*, 2017.

[46] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning Word Vectors for Sentiment Analysis. In *Proceedings of Annual Meeting of the Association for Computational Linguistics: Human language technologies*, 2011.

[47] T. Mihaylova and A. F. Martins. Scheduled Sampling for Transformers. *arXiv preprint arXiv:1906.07651*, 2019.

[48] D. Misra, J. Langford, and Y. Artzi. Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2017.

[49] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *Proceedings of International Conference on Robotics and Automation*, 2018.

[50] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[51] K. Narasimhan, T. Kulkarni, and R. Barzilay. Language Understanding for Text-based Games using Deep Reinforcement Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2015.

[52] K. Nguyen, H. Daumé III, and J. Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. *arXiv preprint arXiv:1707.07402*, 2017.

[53] OpenAI. https://openai.com/blog/chatgpt, 2023.

[54] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training Language Models to Follow Instructions with Human Feedback. *In Proceedings of Neural Information Processing Systems*, 2022.

[55] R. Y. Pang and H. He. Text Generation by Learning from Demonstrations. In *Proceedings of International Conference on Learning Representations*, 2021.

[56] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of Association for Computational Linguistics*, 2002.

[57] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

[58] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[59] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. 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.

[60] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

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

[62] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li. Deep Reinforcement Learning-Based Image Captioning With Embedding Reward. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[63] S. Ross and J. A. Bagnell. Reinforcement and Imitation Learning via Interactive No-regret Learning. *arXiv preprint arXiv:1406.5979*, 2014.

[64] S. Ross, G. Gordon, and D. 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.

[65] T. Salimans and R. Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[66] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[67] J. Scheurer, J. A. Campos, T. Korbak, J. S. Chan, A. Chen, K. Cho, and E. Perez. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*, 2023.

[68] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust Region Policy Optimization). In *Proceedings of International Conference on Machine Learning*, 2015.

[69] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[70] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*, 2015.

[71] J. Skalse, N. H. Howe, D. Krasheninnikov, and D. Krueger. Defining and Characterizing Reward Hacking. *arXiv preprint arXiv:2209.13085*, 2022.

[72] C. Snell, I. Kostrikov, Y. Su, M. Yang, and S. Levine. Offline RL for Natural Language Generation with Implicit Language Q Learning. *arXiv preprint arXiv:2206.11871*, 2022.

[73] A. Sokolov, S. Riezler, and T. Urvoy. Bandit structured prediction for learning from partial feedback in statistical machine translation. *arXiv preprint arXiv:1601.04468*, 2016.

[74] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. Learning to Summarize with Human Feedback. *In Proceedings of Neural Information Processing Systems*, 2020.

[75] T. R. Sumers, M. K. Ho, R. D. Hawkins, K. Narasimhan, and T. L. Griffiths. Learning rewards from linguistic feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6002–6010, 2021.

[76] W. Sun, J. A. Bagnell, and B. Boots. Truncated Horizon Policy Search: Combining Reinforcement Learning & Imitation Learning. *arXiv preprint arXiv:1805.11240*, 2018.

[77] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply Aggrevated: Differentiable Imitation Learning for Sequential Prediction. In *Proceedings of International Conference on Machine Learning*, 2017.

[78] A. Tavakoli, V. Levdik, R. Islam, C. M. Smith, and P. Kormushev. Exploring restart distributions. *arXiv preprint arXiv:1811.11298*, 2018.

[79] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based Image Description Evaluation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[80] A. Venkatraman, M. 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.

[81] S. Wiseman and A. M. Rush. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2016.

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

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

[84] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating Text Generation with Bert. *arXiv preprint arXiv:1904.09675*, 2019.

[85] W. Zhang, Y. Feng, F. Meng, D. You, and Q. Liu. Bridging the gap between training and inference for neural machine translation. *arXiv preprint arXiv:1906.02448*, 2019.

[86] X. Zhang, A. Bosselut, M. Yasunaga, H. Ren, P. Liang, C. D. Manning, and J. Leskovec. Greaselm: Graph reasoning enhanced language models. In *Proceedings of International Conference on Learning Representations*, 2022.

[87] X. Zhang and M. Lapata. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2017.

[88] V. Zhong, C. Xiong, and R. Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

[89] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy. Lima: Less is more for alignment, 2023.

[90] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. 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. (39; 36; 27) propose alignment ideas the attempt to minimize various divergence inspired from maximize entropy RL (30; 31) and Distributional Policy Gradient (DPG) (9). 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 (89; 18).

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 (78). 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 (78) considers incorporating priories to help decide how to prioritize past visited states to incorporate into the initial state distribution. (2) 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(57; 65; 24; 25).

Imitation Learning for Structured Prediction: Many learning-to-search (L2S) approaches have been proposed in the literature that take advantage of IL. One of the first L2S approaches SEARN (22) and DAgger (64) reduce structured prediction tasks to classification. Aggregate Values to Imitate (AggreVate) builds on these algorithms and instead reduces structured prediction to cost-sensitive classification (63). (77) presented a modification of AggreVate, called AggreVateD which is more practical for deep learning applications. (15) proposed a generalization of AggreVate, called LOLS (Locally Optimal Learning to Search) that instead of rolling out with the reference policy, uses a mixture of current policy and reference policy. Under some strong condition, LOLS may yield improved guarantees over AggreVate. These algorithms have inspired algorithms such as Schedule Sampling (SS) (10), methods using SS (23; 47; 28), SEARNN (41), Bridging the Gap (85), and Mixer (60).

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

Input: π_θ, reference π^g, iterations T, dataset $\mathcal{D} = \left\{(x^i, y^i)\right\}_{i=1}^N$
Input: mixing parameter $\beta_1 \in [0,1]$, mixing parameter $\beta_2 \in [0,1]$, mixing prob α
for t = 0,1,...,T-1 **do**

▷ PPO^{++}
    Rollin with $\beta_1 \pi^g + (1-\beta_1)\pi_\theta^t$ starting from $x \sim \mathcal{D}$
    Rollout with π_θ^t to collect trajectories
    Update $V_\phi^{\pi_\theta^t}$ with trajectories and compute advantage estimates $A^{\pi_\theta^t}$

▷ AggreVaTeD
    Rollin with $\beta_2 \pi_\theta^t + (1-\beta_2)\pi^g$ starting from $x \sim \mathcal{D}$
    Rollout with π^g to collect trajectories
    Update $V_\phi^{\pi^g}$ with trajectories and compute advantage estimates $A^{\pi^g}(s,a)$

▷ Mix Update
    Update π_θ 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 (90; 54; 59). Formally, regularized reward function is defined as: $\hat{R}(s_t, a_t, y) = R(s_t, a_t, y) - \beta \text{KL}\left(\pi(a_t|s_t)||\pi_0(a_t|s_t)\right)$ where $\text{KL}\left(\pi(a_t|s_t)||\pi_0(a_t|s_t)\right) = \left(\log \pi(a_t|s_t) - \log \pi_0(a_t|s_t)\right)$ and β is the KL coefficient that is dynamically adapted (54).

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

Table 4: 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 4, 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 (44) 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 (44) 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.

C.3 IMDB - Algorithm Details

Table 5 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 KL coeff: 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 5: 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

Table 6 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.

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 KL coeff: 0.0
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 6: 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.

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 4: **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 5: **Qualitiative Analysis of CommonGen**

F Additional Results

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

Alg	Semantic Score
BC+PPO	0.767 ± 0.018
AggreVaTeD + PPO	0.863 ± 0.007
BC+PPO^{++}	0.883 ± 0.011
D^{2}LOLS(i.e. AggreVaTeD + PPO†)	$\mathbf{0.896 \pm 0.012}$

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