Attribute Controlled Fine-tuning for Large Language Models: A Case Study on Detoxification

Tao Meng[1,2], **Ninareh Mehrabi**[2], **Palash Goyal**[2], **Anil Ramakrishna**[2], **Aram Galstyan**[2]
Richard Zemel[2], **Kai-Wei Chang**[1,2], **Rahul Gupta**[2], **Charith Peris**[2]
[1] University of California, Los Angeles
[2] Amazon.com, Inc.
{tmeng, kwchang}@cs.ucla.edu, perisc@amazon.com

Abstract

We propose a constraint learning schema for fine-tuning Large Language Models (LLMs) with attribute control. Given a training corpus and control criteria formulated as a sequence-level constraint on model outputs, our method fine-tunes the LLM on the training corpus while enhancing constraint satisfaction with minimal impact on its utility and generation quality. Specifically, our approach regularizes the LLM training by penalizing the KL divergence between the desired output distribution, which satisfies the constraints, and the LLM's posterior. This regularization term can be approximated by an auxiliary model trained to decompose the sequence-level constraints into token-level guidance, allowing the term to be measured by a closed-form formulation. To further improve efficiency, we design a parallel scheme for concurrently updating both the LLM and the auxiliary model. We evaluate the empirical performance of our approach by controlling the toxicity when training an LLM. We show that our approach leads to an LLM that produces fewer inappropriate responses while achieving competitive performance on benchmarks and a toxicity detection task.

1 Introduction

Large language models (LLMs) have demonstrated impressive performance across a variety of tasks which has led to their widespread adoption for a multitude of AI applications. However, they carry the risk of producing inappropriate, unsafe, unfair outputs (Wallace et al., 2019; Sheng et al., 2019; Gehman et al., 2020; Huang et al., 2024) Ideally, LLMs should learn to comply with constraints and policies specified by users. For example, in a user-facing application like a chatbot, LLMs should never generate toxic or offensive responses, nor to divulge sensitive information. While there are several post hoc methods to moderate LLM outputs (Lu et al., 2022; Qian et al., 2022; Markov et al., 2023), they lack an efficient and principled approach to training LLMs to adhere to constraints.

We start by defining a sequence-level oracle as a function that takes an LLM's output and adjudicates whether it satisfies a predefined set of attribute constraints. In practice, the oracle can be a rule-based, model-based, or mixed system (e.g., a classifier that decides whether a sentence is toxic). Given a pre-trained LLM and the oracle, we aim to fine-tune an LLM to achieve the following: 1) **Attribute control:** The LLM output passes the oracle with a high probability. 2) **Utility preservation:** The LLM maintains performance comparable to the original LLM on utility benchmarks. 3) **Training efficiency:** The cost of fine-tuning with attribute control is similar to that of the typical fine-tuning.

While existing approaches can meet some of these criteria, achieving all of them is challenging. For example, filtering training data with the oracle function before fine-tuning (Wang et al., 2022) is a simple and efficient method. However, this approach could be less effective. Taking toxicity control as an example, if we filter out the toxic data from a fine-tuning corpus, in a regular context the model will learn not to generate toxic contents. Nevertheless, it might still be possible to trigger the generation of offensive responses given a toxic prompts, due to the fact that toxic prompts are out-of-distribution in relation to the fine-tuning corpus. Another promising approach is reinforcement learning (RL) considering controlling criteria in the reward function (Snell et al., 2023; Mudgal et al., 2023). However, RL setups tend to be inefficient and require preference data generation which adds significant overhead in comparison to generic fine-tuning.

In this work, we propose a novel solution to training an LLM with a set of attribute constraints. Inspired by the classic idea of constraint-driven learning (Chang et al., 2007) and posterior regularization (Ganchev et al., 2010), we incorporate

Findings of the Association for Computational Linguistics: EACL 2024, pages 13329–13341
November 12-16, 2024 ©2024 Association for Computational Linguistics

constraints as a regularizer in fine-tuning. Specifically, we estimate the closest distribution from the current model that satisfies the constraints and penalize the gap from the current model distribution to this estimated distribution to regularize the LLM during fine-tuning. We iterate through this process to push the LLM closer to the feasible region of generations, making the estimation progressively more accurate.

This iterative fine-tuning process updates the base LLM and regularizer sequentially, causing run time to be significantly longer than the typical fine-tuning. Thus, we parallelize our algorithm by updating the base LLM and regularizer simultaneously based on their status in the last iteration. Empirically, the parallelization achieves the same level of performance compared to sequential fine-tuning, and the time complexity is the same as a typical fine-tuning approach.

To validate the effectiveness of our proposed method, we conduct a case study in detoxification, considering three scenarios involving different datasets. In the first scenario, we fine-tune LLMs on datasets rich in toxic language with an attribute control that prevents the generation of toxic outputs. Our approach successfully passes stress tests and produces responses with lower toxicity compared to all baseline models. In the second scenario, we explore whether the attribute control can retain the utility of the LLM while reducing the toxicity of its responses. Training only on a small dataset will lead to catastrophic forgetting. Therefore, we fine-tune the LLM on a mix of data comprising toxiGen (Hartvigsen et al., 2022) and Wikitext (Merity et al., 2016) datasets with attribute control. Our method demonstrates the best balance between model utility and toxicity management compared to similar techniques.

Finally, we assess whether the LLM can effectively identify toxic content without generating it, a critical skill since the model must recognize toxic elements to avoid producing them. In standard fine-tuning, these goals often conflict: the model learns to identify toxicity through training on a toxic corpus, which paradoxically increases the generation of toxic content. However, our method successfully mitigates the generation of toxic content while maintaining classification performance on par with traditional fine-tuning techniques.

We summarize our **contributions** as follows:

- We provide an efficient and effective solution to the attribute-controlled fine-tuning.
- Empirically, we achieve the current best trade-off between attribute control (measured using toxicity) and utility performance against a suite of baselines.
- We show that our approach enables the model to retain knowledge of the concept of a given attribute and yet selectively choose to avoid generating it. This can not be achieved via generic fine-tuning.

2 Related Work

Prior work exists on the controlled generation problem and it can be divided into two fronts. Solutions that apply at inference time during decoding, and solutions that apply at fine-tuning.

Attribute controlled decoding for LLMs Several methods have been explored to control LLM generation during decoding. Some prominent methods include activation editing (Hernandez et al., 2023; Li et al., 2023) which adjusts the activation vectors in the LLM, and weight editing (Meng et al., 2022a; Ilharco et al., 2023) which adjusts the weights in LLM. Dathathri et al. (2020) (PPLM) leverages an auxiliary model to steer the base LLM distribution. Following this line of work, Krause et al. (2021) (GeDi) and Liu et al. (2021) (DExpert) used contrastive learning as an objective to achieve attribute control during decoding. Yang and Klein (2021) (FUDGE) leveraged an external token-level auxiliary model for their work. This was followed by (Meng et al., 2022b) (NADO) who trained a token-level auxiliary model by decomposing the controlling criteria via optimization and approximation. Zhang et al. (2023) (GeLaTo) leverage a probabilistic circuit to tractably incorporate symbolic constraints.

Our work is inspired by NADO (Meng et al., 2022b), and we take it as a sub-component in our fine-tuning approach. However, our paper significantly differs from NADO. Firstly, NADO presents an inference method that guides generation by reweighting output distributions without updating the model weights of the base model. In contrast, our approach involves fine-tuning the model. We use NADO to estimate the optimal distribution that satisfies the constraints. However, our design of the posterior regularizer, the iterative-updated scheme, and the parallel computing algorithm is novel. Moreover, the effectiveness of regularizing

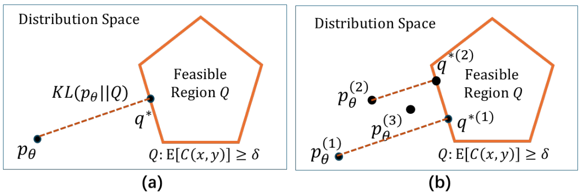

Figure 1: A conceptually visualization of base LLM distribution p_θ and optimal distribution q^* in fine-tuning. The polygon is representing the feasible region Q where the constraints are satisfied. On (a) it shows the regularizer term is defined as the closest distance from p_θ to Q. Regularized by KL-divergence from q, on (b) we show the LLM distribution p_θ is gradually pushed towards the feasible region.

model training with constraints has not been studied for LLMs. We also demonstrate our approach in a real-world application, enhancing models to understand toxicity while preventing the generation of toxic content. This cannot be done by NADO as their approach is only a decoding method.

Attribute controlled fine-tuning for LLMs When controlling attributes during fine-tuning, the most straightforward way is to filter out training data that contain the attribute (Wang et al., 2022). However, this approach tends to cause performance degradation as it can filter out large portions of the training set and does not actively leverage negative examples. Neuro-symbolic approach (Ahmed et al., 2023) incorporates symbolic constraints as loss added to the training objective; however, it cannot handle complex or implicit constraints. RL (Ramamurthy et al., 2023; Snell et al., 2023; Mudgal et al., 2023) can be utilized to control an attribute via the use of an attribute-related reward. RLHF (Ouyang et al., 2022; Xu et al., 2022; Ziegler et al., 2019; Bai et al., 2022a) and RLAIF (Bai et al., 2022b; Lee et al., 2023) leverage feedback from humans and LLMs, respectively, to control the required attributes. However, they are more focused on being aligned with humans (or LLMs) rather than specific attribute control. RL-based methods are effective but often inefficient due to the large variance in feedback provided by the reward models.

3 Methodology

3.1 Notation and Formalization

We use p_θ to denote the LLM and θ is its trainable weights, $\mathbf{x} \in \mathcal{X}$ is the input (e.g. prompts), and $\mathbf{y}$ is the generated output sequence of the model. We denote $\mathbf{y}_{<i} = (y_0, y_1, \ldots, y_{i-1})$ as the prefix of $\mathbf{y}$. $C(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ denotes a black box oracle function which takes prompt $\mathbf{x}$ and model output $\mathbf{y}$ as input, and outputs whether the generation $\mathbf{y}$ satisfies the constraints.[1] For example, in detoxification, the oracle takes a user prompt $\mathbf{x}$ and the model response $\mathbf{y}$ as input, then returns 0 when the response is offensive, indicating that the response is unacceptable.

Given an LLM p_θ, a black box oracle $C(\mathbf{x}, \mathbf{y})$, and a training dataset $D = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, our goal is to fine-tune the LLM as $p_{\tilde{\theta}}$ so that the model retains its utilities while satisfying the constraints in expectation:

$$\forall \mathbf{x} \in \mathcal{X},\ \mathbb{E}_{\mathbf{y} \sim p_{\tilde{\theta}}(\mathbf{y}|\mathbf{x})}[C(\mathbf{x}, \mathbf{y})] \geq \delta. \tag{1}$$

Here, δ is a user-specified parameter. When $\delta = 1$, the training will push the model to satisfy all constraints, while choosing $0 < \delta < 1$, soft constraints are enforced.

3.2 Fine-tuning LLM with Posterior Regularization

Given a training data $(\mathbf{x}, \mathbf{y})$, typically the objective we fine-tune the LLM p_θ is defined as

$$L_{LM}(p_\theta; \mathbf{x}, \mathbf{y}) = \sum\nolimits_i L_{CE}(p_\theta(y_i|\mathbf{x}, \mathbf{y}_{<i}), 1),$$

where L_{CE} is the cross-entropy loss. To achieve attribute control, we propose to add a regularization term that penalizes the violation of constraints.

The general idea of our approach is to fine-tune LM with a regularizer to penalize the following posterior regularization (Ganchev et al., 2010), we define

$$Q := \{q \mid \forall \mathbf{x} \in \mathcal{X}, \mathbb{E}_{\mathbf{y} \sim q(\mathbf{y}|\mathbf{x})}[C(\mathbf{x}, \mathbf{y})] \geq \delta\}$$
$$D_{KL}(p_\theta \| Q) := \min\nolimits_{q \in Q} D_{KL}(p_\theta \| q).$$

The feasible region Q is the set of distributions that satisfy the constraint in Eq. (1). Illustrated by Fig. 1(a), the regularization term D_{KL} is defined as the smallest divergence from Q measured by Kullback–Leibler (KL) divergence. The overall objective of fine-tuning is

$$L(p_\theta; \mathbf{x}, \mathbf{y}, Q) := L_{LM}(p_\theta; \mathbf{x}, \mathbf{y}) + \lambda D_{KL}(p_\theta \| Q), \tag{2}$$

where λ is the hyper-parameter balancing the two terms.

[1] We can extend our approach to handle real value constraints in the form of $C(\mathbf{x}, \mathbf{y}) \in [0, 1]$. For simplicity, we consider binary constraints in this paper.

However, the second term is intractable and hard to compute: when the base model distribution changes in fine-tuning, the closest distribution also changes. To address this issue, we design an iterative fine-tuning process: we first fix the base model distribution and estimate the closest distribution in the feasible set (Sec. 3.3, 3.4), and then we fix the estimated distribution as the reference distribution in the KL regularizer to fine-tune the LLM (Sec. 3.5). To speed up the process, we further propose parallel fine-tuning (Sec. 3.6).

3.3 Optimal Distribution in Feasible Region

To compute the regularizer term in fine-tuning, we need to find the optimal distribution q^* as the reference distribution by solving the following problem

$$q^* = \arg\min_{q:E_{\mathbf{y}\sim q(\mathbf{y}|\mathbf{x})}[C(\mathbf{x},\mathbf{y})]\geq\delta} D_{KL}(q\|p). \tag{3}$$

Meng et al. (2022b) shows the close-form solution can be derived as

$$q^*(y_i|\mathbf{x},\mathbf{y}_{<i}) \propto p_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})\cdot$$
$$[(\delta - R^p_C(\mathbf{x}))R^p_C(\mathbf{x},\mathbf{y}_{<i}\oplus y_i) + (1-\delta)R^p_C(\mathbf{x})]$$

if $\delta > R^p_C(\mathbf{x})$. Otherwise the constraint is already satisfied and $q^*(y_i|\mathbf{x},\mathbf{y}_{<i}) = p_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})$.

Specifically, when $\delta = 1$, we have

$$q^*(y_i|\mathbf{x},\mathbf{y}_{<i}) \propto p_\theta(y_i|\mathbf{x},\mathbf{y}_{<i})R^p_C(\mathbf{x},\mathbf{y}_{<i}\oplus y_i), \tag{4}$$

where $\oplus$ is the concatenation operation. $R^p_C(\mathbf{x},\mathbf{y}_{<i})$ is the probability that the generated output will satisfy constraints when the generation finishes given input $\mathbf{x}$ and prefix $\mathbf{y}_{<i}$, and is given by

$$R^p_C(\mathbf{x},\mathbf{y}_{<i}) = \Pr_{\mathbf{y}\sim p_\theta(\mathbf{y}|\mathbf{x},\mathbf{y}_{<i})}[C(\mathbf{x},\mathbf{y})=1],$$
$$R^p_C(\mathbf{x}) = \Pr_{\mathbf{y}\sim p_\theta(\mathbf{y}|\mathbf{x})}[C(\mathbf{x},\mathbf{y})=1].$$

Basically, the satisfaction probability R^p_C is the token-level decomposition of the sentence-level oracle C. Based on δ and R^p_C, the solution shows how to adjust the next token distribution from the original distribution p_θ.

Unfortunately, although the function R^p_C is well-defined, it is not tractable. To achieve the optimal solution in Eq. (3), in this work, we estimate R^p_C from the training data and the LLM, and update the two terms in Eq. (3) iteratively. In sections 3.4 and 3.5, we describe how we estimate R^p_C from the data and the current model p_θ, and how we update the model p_θ with the help of the estimated R^p_C.

Note that in fine-tuning objective Eq. (2), the reference distribution q is fixed, and we update p_θ, so the regularizer is $D_{KL}(p_\theta\|q)$. However, here the model p is fixed and we seek the optimal q, so we minimize $D_{KL}(q\|p_\theta)$. Empirically, when we optimize the KL divergence term, the trainable weights in the reference distribution usually lead to unstable training. Thus, we always set the fixed distribution as the reference distribution.

3.4 Estimating R^p_C from LLM and Data

To estimate R^p_C, we train an auxiliary model R_ϕ from the training data $\tilde{D}$ weighted by the base LLM p_θ. We assume the empirical distribution is drawn from unseen training distribution D, and has no repetition[2], and set the objective function for a particular example $(\mathbf{x},\mathbf{y}) \in \tilde{D}$ as the cross-entropy loss between the predicted satisfaction probability and oracle output, weighted by the sequence probability $p_\theta(\mathbf{y}|\mathbf{x})$

$$\begin{aligned}&L(R_\phi;\mathbf{x},\mathbf{y})\\ =&p_\theta(\mathbf{y}|\mathbf{x})\textstyle\sum_i L_{CE}(R_\phi(\mathbf{x},\mathbf{y}_{<i}),C(\mathbf{x},\mathbf{y}_{<i})).\end{aligned} \tag{5}$$

Considering the expected loss on distribution D, we have

$$\begin{aligned}&\mathbb{E}_{\tilde{D}\sim D,(\mathbf{x},\mathbf{y})\sim\tilde{D}}[L(R_\phi;\mathbf{x},\mathbf{y})]\\ =&\mathbb{E}_{\tilde{D}\sim D,(\mathbf{x},\mathbf{y})\sim\tilde{D}}[p_\theta(\mathbf{y}|\mathbf{x})\sum_i L_{CE}(R_\phi(\mathbf{x},\mathbf{y}_{<i}),C(\mathbf{x},\mathbf{y}))]\\ =&\mathbb{E}_{\mathbf{x}\sim D,\mathbf{y}\sim p_\theta(\mathbf{y}|\mathbf{x})}[\sum_i L_{CE}(R_\phi(\mathbf{x},\mathbf{y}_{<i}),C(\mathbf{x},\mathbf{y}))]\\ =&\sum_i L_{CE}(R_\phi(\mathbf{x},\mathbf{y}_{<i}),R^p_C(\mathbf{x},\mathbf{y}_{<i})).\end{aligned} \tag{6}$$

Therefore, the global minimum of the expected loss function is reached when $R_\phi(\mathbf{x},\mathbf{y}_{<i}) = R^p_C(\mathbf{x},\mathbf{y}_{<i})$.

In Meng et al. (2022b) the auxiliary model is trained by the data sampled from p_θ without weighting the data by its probability as Eq. (5). The expected loss is the same as Eq. (6). In our experiments, we apply sampling to train the auxiliary model, when there is no available training data. Hereafter in this work, we follow Meng et al. (2022b) and refer to this auxiliary model as the neurally-decomposed oracle (NADO). In practice, NADO architecture is similar as the base LLM, with the same hidden dimension and fewer layers.

[2] If there are repeated examples we can remove them before training. This assumption makes sure that the following weighted empirical loss mimics the expectation loss from sampling.

3.5 Iteratively Updating p_θ by Regularized Fine-tuning

Once we estimate R_p^C by NADO R_ϕ, we are able to get the estimated optimal distribution q from Eq. (3) by replacing R_p^C with R_ϕ. We then plug in the estimated optimal distribution to the fine-tuning objective in Eq. (2) as

$$\begin{aligned}&L(p_\theta; \mathbf{x}, \mathbf{y}, q)\\ =&L_{LM}(p_\theta; \mathbf{x}, \mathbf{y}) + \lambda D_{KL}(p_\theta(\mathbf{y}|\mathbf{x}) \| q(\mathbf{y}|\mathbf{x}))\\ =&\sum\nolimits_i \log p_\theta(y_i|\mathbf{x}, \mathbf{y}_{<i})\\ &+\lambda D_{KL}(p_\theta(y_i|\mathbf{x}, \mathbf{y}_{<i}) \| q(y_i|\mathbf{x}, \mathbf{y}_{<i})).\end{aligned} \tag{7}$$

Intuitively, a model fine-tuned with the objective in Eq. (7) exhibits a trade-off between the model quality and the amount of control. Fine-tuned on this objective, the model converges at some mid-point between p_θ and q.

Now we are able to estimate R_p^C by R_ϕ from the training data and model p_θ (Sec. 3.4), and fine-tune p_θ with estimated optimal distribution q derived from R_ϕ. A straightforward way is to update these models iteratively, which we call "sequential fine-tuning". In this process, we gradually push the base model distribution towards the feasible region, and the estimated optimal distribution is more accurate. As shown in Fig. 2(a) and described in Sec. 3.1, we iteratively run the following three steps:

- Based on current LLM $p_\theta^{(i)}$, sample or weight data $D^{(i)}$ labeled by the oracle.
- Train NADO $R_\phi^{(i)}$ using the data $D^{(i)}$ initialized with $R_\phi^{(i-1)}$.
- Fine-tune the LLM $p_\theta^{(i)}$ with the KL-divergence between $p_\theta^{(i)}$ and $q^{(i)}$ given by Eq. (4).

The distribution of the base model can be conceptually visualized in Fig. 1(b) during fine-tuning. As the base model $p_\theta^{(i)}$ getting closer to the feasible region, the estimated optimal distribution $q^{(i)}$ will be more accurate compared to the estimation from the original base model distribution $q^{(1)}$.

3.6 Parallel Fine-tuning

The iterative fine-tuning process outlined in Section 3.5 executes its steps sequentially by solving the optimization problem in Eq. (3) in each round. However, while accurate, it is also inefficient. In this section, we propose a parallel fine-tuning method to improve efficiency.

In parallel fine-tuning, we propose a set-up that processes the three steps outlined in Sec. 3.5 in parallel (see Fig. 2(b)). Given $p_\theta^{(i)}$, $D^{(i)}$ and $q^{(i)}$, the following three steps are processed simultaneously:

- Based on current LLM $p_\theta^{(i)}$, sample or weight data $D^{(i+1)}$ labeled by the oracle.
- Train NADO $R_\phi^{(i+1)}$ using data $D^{(i)}$ initialized with $R_\phi^{(i)}$.
- Fine-tune the LLM $p_\theta^{(i+1)}$ with the KL-divergence from $q^{(i)}$ given by Eq. (4).

After one round, we get $p_\theta^{(i+1)}$, $D^{(i+1)}$ and $q^{(i+1)}$. The LLM keeps fine-tuning on the dataset with a regularizer, and the regularizer is updated at every checkpoint. In sequential fine-tuning, the process will terminate at each checkpoint, waiting for the regularizer to update with the data sampled or weighted by the LLM. Compared to a baseline, which fine-tunes without control, the extra time cost in our method is only the extra computation on the regularizer and the time cost in dumping checkpoints. The additional memory cost for NADO is not significant, because it is relatively small compared to the base LLM.

In practice, we select proper hyperparameters[3] to ensure the three steps take similar computational time. In such a case, parallel fine-tuning achieves 3x speed up compared to sequential fine-tuning.

3.7 Adaptive Regularizer

The data for fine-tuning an LLM often includes a diverse mix of sources. Fine-tuning on a specific domain may lead to performance degradation in other domains due to catastrophic forgetting. A popular approach is to add KL-divergence to the original model to avoid the model deviating from the original model (Schulman et al., 2017). To effectively incorporate this mechanism into our approach, we can implement domain-specific regularizers during the fine-tuning process.

Specifically, we denote the training dataset as $D = \bigcup_i D_i$. For each subset D_i with a corresponding constraint oracle C_i. We use the base LLM to weight the subset, and C_i to label them. According to Eq. (5), we train NADO R_{ϕ_i} for the constraint oracle C_i, and compute the estimated optimal distribution q_i. We use q_i as the reference distribution in the KL-divergence. Specifically, when we set q_i as the original distribution, the regularizer sets as

[3] Including the number of examples we sample or weight, number of epochs to fine-tune LM, and number of epochs to train NADO.

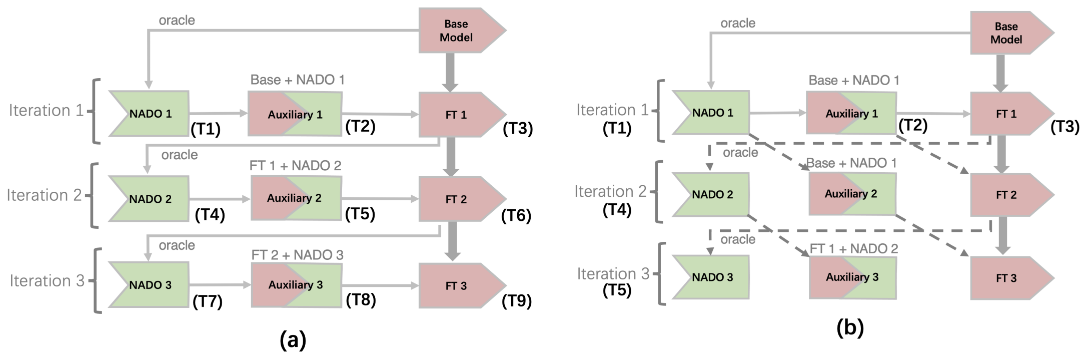

Figure 2: An illustration of sequential and parallel fine-tuning for three iterations. We use T (time step) to indicate the time. Oracle, symbolizes the process of sampling data from an LLM, labeling with an oracle, and training the NADO model. On the left, we show sequential execution with the grey arrows showing the direction of flow. On the right, we show the parallelized execution. Note that in this case, all components (left to right) of each iteration are run at the same time step (except in iteration 1). Note also, that the grey dashed arrows (from iteration 2 onwards) do not flow across components within the same iteration level, indicating the independence of each component from other components in the same level. This allows them to be executed in parallel.

the KL-divergence to the original model. We refer to this regularizer as the *preserving regularizer*.

In this work, we demonstrate how to effectively control the toxicity of an LLM while preserving its performance level. We apply the regularize for toxicity control when fine-tuning it on toxicity-related datasets, while using the preserving regularize on a general dataset (like Wikitext). Formally, we denote p_0 as the original LLM, q as the estimated optimal distribution under toxicity constraint oracle, and $D_T \subset D$ as the toxicity-related training set. We adopt the fine-tuning objective in Eq. (2) to

$$\begin{aligned} L(p_\theta; D, q) &= \sum\nolimits_{(\mathbf{x},\mathbf{y})\in D} L_{LM}(p_\theta; \mathbf{x}, \mathbf{y}) \\ &+\lambda \sum\nolimits_{(\mathbf{x},\mathbf{y})\in D_T} D_{KL}(p_\theta(\mathbf{y}|\mathbf{x}) \| q(\mathbf{y}|\mathbf{x})) \\ &+\lambda \sum\nolimits_{(\mathbf{x},\mathbf{y})\notin D_T} D_{KL}(p_\theta(\mathbf{y}|\mathbf{x}) \| p_0(\mathbf{y}|\mathbf{x})). \end{aligned} \tag{8}$$

4 Case Study on Detoxification

To test the effectiveness of the proposed approach, we apply it to detoxify an LLM. Toxicity, as discussed in Section 1, is of significant importance as a metric for the evaluation of LLM (Brown et al., 2020; Touvron et al., 2023a; Chowdhery et al., 2022; Touvron et al., 2023b). In this context, we apply our fine-tuning schema in three different scenarios; (1) **detoxification:** testing the effectiveness of our proposed approach in attribute control, (2) **multi-task scenario:** testing that the controlled model preserves the same level performance on other tasks, and (3) **toxicity classification:** testing whether the control affects the model performance on attributes related tasks.

Model	API Tox.	ToxiGen
Llama baseline	0.315	23.0
Reinforcement Learning	0.269	12.3
NADO Decoding Control	0.289	14.4
Ours (sequential)	**0.259**	11.0
Ours (parallel)	0.261	**10.9**

Table 1: Toxicity scores of Llama-7B model with different detoxification methods. The proposed fine-tuning methods outperform RL and decoding-time control in detoxification. Parallel fine-tuning achieves similar control compared to sequential, with 3x fine-tuning speed.

In all experiments, we set $\delta = 1$ to set toxicity as a hard constraint. Detailed notes on data pre-processing, hyper-parameter choice for model training, and the architecture of auxiliary models can be found in the Appendix.

4.1 Detoxification

Given a corpus and a toxicity oracle, we first show the effectiveness of our approach in detoxification. We also show that parallel fine-tuning achieves a similar performance as the sequential one.

Setup We use Llama-7B (Touvron et al., 2023a) as the base model. NADO has a similar architecture but with only 8 layers. For our experiments, we use RealToxicPrompts (RTP) (Gehman et al., 2020) and ToxiGen (Hartvigsen et al., 2022) datasets. For each dataset, we sample 50k prompts for fine-tuning and another 5k for evaluation. During the evaluation, we prompt the model with each data point from the evaluation set and generate 32 tokens. For the RTP dataset, we measure the average toxicity across the generations by using PerspectiveAPI. For ToxiGen, we use the pre-trained Toxigen (RoBERTa) classifier, which was released with the dataset, to calculate the percentage of generated sentences that are toxic. We test three detoxification methods, in addition to the Llama baseline:

- **Reinforcement Learning:** For each prompt in the evaluation set, we sample 32 generations. We utilize the PerspectiveAPI and ToxiGen classifier confidence scores as reward for the two test sets, respectively. We then use the policy gradient (Sutton et al., 1999) to update the base language model.
- **NADO controlled decoding:** For each prompt in the two test sets, we sample 32 sentences and obtain binary labels from PerspectiveAPI and the ToxiGen classifier, respectively. When using PerspectiveAPI we set a threshold of toxicity score > 0.1 as *toxic*.
- **Ours:** We follow the NADO-controlled decoding oracle setup. We split the 50k fine-tuning set into 5 groups. We separately run iterative sequential fine-tuning and parallel fine-tuning for 5 rounds using these groups.

Results The results are shown in Tab. 1. We observe that on both datasets our method achieves the best detoxification (given the same amount of training data). We observe that there is a significant performance improvement brought on by iterative fine-tuning when compared to NADO-controlled decoding, which shows that directly estimated the optimal distribution is not optimal. The iterative process enables the gradual push of the base model distribution towards the feasible region (Fig 1), and the estimated optimal distribution improves in its accuracy. The sequential and parallel fine-tuning results show comparable performance. Since parallel fine-tuning is more efficient, we focus on this method from this point onward.

Preference optimization (Rafailov et al., 2023) is a popular RL method in fine-tuning LLMs. It leverages human preference between a pair of generation to achieve an alignment between model output and human. However, in our setup, the goal is to control the toxicity of the model output. The metric is clearly defined by PerspectiveAPI or ToxiGen classifier. Directly applying the toxicity value as the reward in RL is much more effective than the pairwise preference.

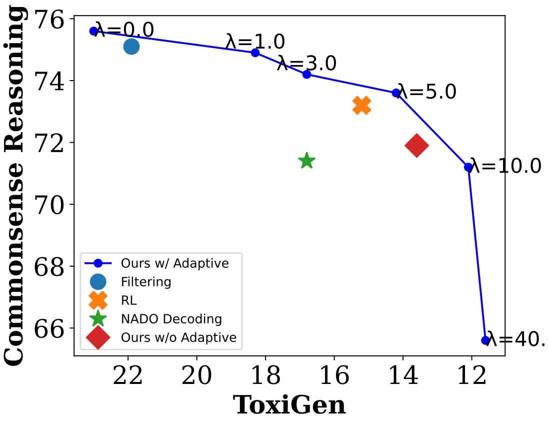

Figure 3: The trade-off curve between ToxiGen performance and Commonsense reasoning performance when fine-tuning Llama-7B model with our proposed approach with adaptive regularizer, compared to the listed baselines in Table 2. The trade-off is controlled by the coefficient λ in Eq. (8). We observe that to control the language model in the same level of toxicity, our approach, with adaptive regularizer, achieves the best commonsense reasoning performance compared to the listed methods.

4.2 Balance between Utility and Detoxification

We further study how our method can reduce toxicity generation while maintaining model utility. As RTP and ToxiGen datasets are small, fine-tuning only on them would lead to catastrophic forgetting and degradation in utility. Therefore, we fine-tune the LLM on a mix of general Wikitext corpus and toxicity corpus. We show that the proposed method achieves the best trade-off between toxicity control and maintaining performance on general utility benchmarks.

Setup We use Llama-7B (Touvron et al., 2023a) and Falcon-7B (Almazrouei et al., 2023) as base models, and fine-tune each of them on a mixture of ToxiGen and Wikitext (Merity et al., 2016) data in equal proportions. We evaluate model performance on ToxiGen toxicity, and the utility on MMLU and commensense reasoning. The details about evaluation metrics can be found in the Appendix. We test 5 different methods:

- **Filtering:** We filter out all the data labeled as *toxic* by the ToxiGen classifier.
- **Reinforcement Learning:** We take the confidence score provided by the ToxiGen classi-

Model		ToxiGen	MMLU(5-shot)	Com. Reasoning (0-shot)
Llama-7B	Baseline	23.0	35.1	75.6
	Filtering	21.9	34.6	75.1
	RL	15.2	33.6	73.2
	NADO decoding	16.8	31.1	71.4
	Ours w/o Adaptive	13.6	30.4	71.9
	Ours w/ Adaptive	14.2	33.9	73.6
Falcon-7B	Baseline	14.0	27.2	76.1
	Filtering	13.6	26.4	74.9
	RL	9.8	25.4	74.4
	NADO decoding	7.3	23.6	72.5
	Ours w/o Adaptive	7.1	24.1	71.8
	Ours w/ Adaptive	7.3	26.1	74.5

Table 2: Benchmark performance of Llama-7B and Falcon-7B with toxicity control. The models are fine-tuned on a mixture corpus including ToxiGen and Wikitext in equal proportions. Results show that our approach achieves a better trade-off between toxicity control and benchmark performance compared to RL. Filtering is not effective in controlling toxicity. With the adaptive regularizer, LLM has a significant performance improvement on benchmarks.

fier as the reward, and apply policy-gradient to minimize the toxicity.

- **NADO decoding:** We train the auxiliary model on ToxiGen sampled data, and control the model generation at decoding time.

- **Ours (without Adaptive):** We apply parallel fine-tuning on both datasets with the auxiliary model trained on ToxiGen sampled data.

- **Ours (with Adaptive):** We apply an adaptive regularizer as described in Eq. (8). We use the preserving regularizer on Wikitext data, while using the toxicity control regularizer on the ToxiGen sampled data.

Results The results are shown in Tab. 2. We observe that all detoxification methods cause a performance drop on our utility metrics (i.e. MMLU and commonsense reasoning). Filtering is not effective for detoxification. In Fig. 3 we show the trade-off curve between ToxiGen and Commonsense reasoning tasks of our method compared to other methods. Our method with the adaptive regularizer achieves the best trade-off between toxicity control and model utility.

When used without the adaptive regularizer, our method achieves the best toxicity control. However, this comes at the cost of utility loss. This indicated that the toxicity regularizer trained on ToxiGen sampled data does not perform well on the Wikitext data. The adaptive regularizer helps preserve the model utility while fine-tuning on Wikitext data.

Win rate	Base	Filter	RL	Ours
Base	N/A	44.3	45.1	51.4
Filter	55.7	N/A	53.4	61.6
RL	54.9	46.6	N/A	61.3
Ours	48.6	38.4	38.7	N/A

Table 3: Pairwise comparison by OPT-30B on ToxiGen sampling data. The value shows the win rate of the method on the top row in pairwise comparison. Our model is indistinguishable from base model and outperforms Filter and RL approaching, demonstrating it retains the quality of generation.

We note that Falcon-7B has much lower toxicity when compared to Llama-7B. The consistent performance trends observed in both base models, demonstrate that our method is robust to different base models independent of its levels of toxicity.

We further analyze model generation quality by leveraging a larger model, OPT-30B (Zhang et al., 2022), to do pairwise comparison on model generations for ToxiGen prompts from 4 systems: (1) the base Llama-7B model, (2) filtering, (3) RL and (4) ours with the adaptive regularizer. We do not consider NADO controlled decoding and ours without the adaptive regularizer, as they are obviously worse in terms of model quality. The results are shown in Tab. 3. We show that OPT-30B prefers our system (with the adaptive regularizer) the best, with slight improvement over the base model.

Model	API Tox.	Classify ROC
baseline	0.315	0.910
SFT(LLM loss)	0.344	**0.966**
Ours(LLM loss)	**0.288**	0.959
SFT(classification)	0.314	0.972

Table 4: Jigsaw dataset performance of Llama-7B model with toxicity control. SFT with LLM loss shows a trade-off between the generation toxicity and classification performance, while our approach is capable to reduce the generation toxicity while improve toxicity classification performance.

4.3 Toxicity Classification and Generation

An LLM cannot avoid generating toxic outputs if it is unable to recognize toxic language. Therefore, it is essential for LLM to comprehend the characteristics of toxic content so that it can actively filter out harmful elements while maintaining the integrity of the generated output. However, a generic fine-tuning method often cannot improve the toxicity classification and reduce toxic generation at the same time. We design an experiment to test whether our approach can effectively enhance the LLM's ability to classify toxic content without increasing its generation of such content.

Setup We fine-tune the Llama-7B on the Jigsaw toxicity classification dataset (Jain et al., 2022). We compare the performance of models fine-tuned using our controlled method to ones fine-tuned using uncontrolled fine-tuning. We use classification performance and generation toxicity (as evaluated by PerspectiveAPI) as metrics of comparison. Specifically, we compare three methods:

- **Supervised fine-tuning with LLM loss:** We concatenate each question and answer in the Jigsaw dataset, and fine-tune with a language modeling objective.
- **Ours with LLM loss:** We train an auxiliary model on RTP sampled data labeled by PerspectiveAPI, and fine-tune the language model same as above on Jigsaw dataset with the toxicity regularizer.
- **Supervised fine-tuning as classification:** We treat each question in Jigsaw as the prompt and only calculate loss on the answers. This is regarded the upper bound of performance for this task.

Results The results are shown in Tab. 4. We observe that if we fine-tune the LLM on the Jigsaw dataset without toxicity control, the generation toxicity increases significantly (9.2%, from 0.315 to 0.344). The reason is that Jigsaw consists of toxic content and fine-tuning on this shifts the model out distribution to be toxic. In comparison, when using our fine-tuning schema which leverages the toxicity regularizer, we achieve decreased toxicity. Notably our method also improves classification performance, achieving almost similar performance to uncontrolled fine-tuning, demonstrating our approach makes the model understand the toxicity rather than simply make model ignore the toxicity contents in training data.

5 Conclusion

We propose a novel fine-tuning approach for attribute control in LLM generations and we demonstrate its effectiveness using toxicity as our chosen attribute. While this work focuses on toxicity, our approach is general enough to accommodate other types of attributes as well. With adaptive regularizers, our method can further extend to control multiple attribute across various domains.

Limitation

In this work we assume that a decent oracle (PerspectiveAPI, ToxiGen classifier) for the attribute we would like to control is available. A low quality oracle may rely on superficial shortcut between generation and constraint labels, resulting in that the fine-tuned model captures such shortcut. Therefore, it is possible that we need to train a decent oracle before applying the proposed method.

Although our method is general to apply different kinds of constraints since we have no assumption on the black-box oracle, in experiment we focus on detoxification. We leave the study on controlling other attributes in future work.

As a attribute control method, we note that there is a potential risk that malicious users could use this approach to 'toxify' the LLM by opposite the oracle. In addition, the generated texts may contain new toxic contents that cannot be generated in original LLM, since it may learn from the toxic fine-tuning corpus. However, on the other hand, the controlled LLM is generally less risky in generating toxic contents.

References

Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. 2023. A pseudo-semantic loss for autoregressive models with logical constraints. In *NeurIPS*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *CoRR*, abs/2311.16867.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022b. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *AAAI*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*. The Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT (1)*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *ICLR*. OpenReview.net.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularizaftion for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 3356–3369. Association for Computational Linguistics.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *ACL (1)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023. Measuring and manipulating knowledge representations in language models. *CoRR*, abs/2304.00740.

Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Hanchi Sun, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bertie Vidgen, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao, Chunyuan Li, Eric P. Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, Joaquin Vanschoren, John Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Yang Wang, Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, Yong Chen, and Yue Zhao. 2024. Trustllm: Trustworthiness in large language models. In *Forty-first International Conference on Machine Learning*.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *ICLR*. OpenReview.net.

Naman Jain, Skanda Vaidyanath, Arun Shankar Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram K. Rajamani, and Rahul Sharma. 2022. Jigsaw: Large language models meet program synthesis. In *ICSE*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *EMNLP (Findings)*, pages 4929–4952. Association for Computational Linguistics.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. RLAIF: scaling reinforcement learning from human feedback with AI feedback. *CoRR*, abs/2309.00267.

Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *NeurIPS*.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL/IJCNLP (1)*.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. Neurologic a*esque decoding: Constrained text generation with lookahead heuristics. In *NAACL-HLT*, pages 780–799. Association for Computational Linguistics.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15009–15018.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. In *NeurIPS*.

Tao Meng, Sidi Lu, Nanyun Peng, and Kai-Wei Chang. 2022b. Controllable text generation with neurally-decomposed oracle. In *NeurIPS*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. 2023. Controlled decoding from language models. *CoRR*, abs/2310.17022.

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

Jing Qian, Li Dong, Yelong Shen, Furu Wei, and Weizhu Chen. 2022. Controllable natural language generation with contrastive prefixes. In *ACL (Findings)*. Association for Computational Linguistics.

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

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *ICLR*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *EMNLP/IJCNLP (1)*, pages 3405–3410. Association for Computational Linguistics.

Charlie Snell, Ilya Kostrikov, Yi Su, Sherry Yang, and Sergey Levine. 2023. Offline RL for natural language generation with implicit language Q learning. In *ICLR*.

Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063. The MIT Press.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *EMNLP/IJCNLP (1)*, pages 2153–2162. Association for Computational Linguistics.

Boxin Wang, Wei Ping, Chaowei Xiao, Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Bo Li, Anima Anandkumar, and Bryan Catanzaro. 2022. Exploring the limits of domain-adaptive training for detoxifying large-scale language models. *CoRR*, abs/2202.04173.

Jing Xu, Megan Ung, Mojtaba Komeili, Kushal Arora, Y-Lan Boureau, and Jason Weston. 2022. Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback. *CoRR*, abs/2208.03270.

Kevin Yang and Dan Klein. 2021. FUDGE: controlled text generation with future discriminators. In *NAACL-HLT*. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *ACL (1)*.

Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. 2023. Tractable control for autoregressive language generation. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 40932–40945. PMLR.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593.

A Model Architecture and Optimizer

In all experiments, the NADO model has the same configuration as Llama-7B model but with only 8 layers. We use AdamW as optimizer with learning rate $3e-5$ and weight decay $1e-2$.

In detoxification and toxicity classification experiments, we train NADO from data sampled by base LLM. We use simple random sampling without any decoding configuration.

B LLM Fine-tuning

We fine-tune the base LLM with AdamW optimizer with learning rate $1e-5$ and weight decay $1e-2$. $\lambda = 10.0$ in the multi-task scenario experiment, and $\lambda = 5.0$ in the detoxification and toxicity classification experiments.

C Metrics in Multitask Experiment

We evaluate the model performance on the following three metrics:

- **ToxiGen (toxicity):** Same set up as the detoxification experiment in Section 4.1.
- **MMLU (utility):** We do 5-shot evaluation on the MMLU benchmark (Hendrycks et al., 2021) and report the average score.
- **Commonsense Reasoning (utility):** We do 0-shot evaluation on 4 commonsense reasoning benchmarks, BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019) and WinoGrande (Sakaguchi et al., 2020), and report the average score.

D Data Preprocessing

RTP and ToxiGen: We randomly select prompts, and use the LLM to randomly sample 32 tokens in both training and evaluation.

Jigsaw: The data are comment-label pairs. We templatize the data as:

The comment *[comment]* is a *[label name]* comment.

In evaluation, we query the model by template:

Is the comment *[comment]* a *[label name]* comment? Answer: *[Yes / No]*

MMLU and commensense reasoning: We follow the standard o-shot and few-shot evaluation scripts.

E License of Datasets

The licenses of datasets we use in this paper list below:
ToxiGen: MIT License
Wikitext: CC BY-SA License and GFDL License
Jigsaw: MIT License
MMLU: GNU AGPL
BoolQ: CC BY-SA License
PIQA: Apache License
HellaSwag: MIT License
WinoGrande: Apache License