Continual Reinforcement Learning for Controlled Text Generation

Velizar Shulev, Khalil Sima'an
Institute for Logic, Language and Computation
University of Amsterdam
velizar.shulev@gmail.com, k.simaan@uva.nl

Abstract

Controlled Text Generation (CTG) steers the generation of continuations of a given context (prompt) by a Large Language Model (LLM) towards texts possessing a given attribute (e.g., topic, sentiment). In this paper we view CTG as a Continual Learning problem: how to learn at every step to steer next-word generation, without having to wait for end-of-sentence. This continual view could be useful for online applications such as CTG for speech, where end-of-sentence is often uncertain. We depart from an existing model, the Plug-and-Play language models (PPLM), which perturbs the context at each step to better predict next-words that posses the desired attribute. While PPLM is intricate and has many hyper-parameters, we provide a proof that the PPLM objective function can be reduced to a Continual Reinforcement Learning (CRL) reward function, thereby simplifying PPLM and endowing it with a better understood learning framework. Subsequently, we present, the first of its kind, CTG algorithm that is fully based on CRL and exhibit promising empirical results.

Keywords: topic control, controlled text generation, continual reinforcement learning

1. Introduction

Work on Controlled Text Generation (CTG) aims at directing the generation by a large language model (LLM) towards texts containing a desired attribute, such as topic, sentiment or domain. Given an LLM distribution over text $p(\mathbf{x})$, CTG can be seen to condition the generated text on a given attribute a, i.e., $p(\mathbf{x}|a)$. One CTG approach involves fine-tuning on the attribute using class-conditional LMs (Keskar et al., 2019; Ziegler et al., 2019; Krause et al., 2021; Arora et al., 2022). This can be computationally expensive, requiring fine-tuning for each attribute. In contrast, weighted-decoding algorithms aim to avoid intensive computation by re-weighing the LLM probability $p(\mathbf{x})$ (Holtzman et al., 2018; Ghazvininejad et al., 2017; Dathathri et al., 2020; Yang and Klein, 2021). This is often done using Bayes' inversion, $p(\mathbf{x}|a) \propto p(a|\mathbf{x})\ p(\mathbf{x})$, to sample from the *original* LLM ($p(\mathbf{x})$) re-weighed using a lightweight *discriminator model* $p(a|\mathbf{x})$. This leaves the LLM untouched and concentrates on devising and combining discriminator models with the LLM. This work falls squarely within the latter framework.

In this paper we view CTG as a Continual Learning problem: at each generation time step, the algorithm learns directly how to steer the next-word towards the desired attribute. This is useful for online application such as CTG for speech, where end-of-sentence is often uncertain. We start out from the Plug-and-Play Language Models (PPLM) approach (Dathathri et al., 2020), which stands out as continuously manipulating the context at each generation step. PPLM is intricate, heuristically interpolating multiple models and possessing many hyper-parameters. We first prove that the PPLM objective can be cast as a Continual Reinforcement Learning (CRL) reward function (Abel et al., 2023). Subsequently we propose PPO$_{LM}$[1], a single hyper-parameter CRL algorithm based on Proximal Policy Optimization (PPO) (Schulman et al., 2017) – a state-of-the-art policy gradient RL method, and show improved empirical results relative to PPLM. Our work is the first applying CRL to CTG, but also to LLMs in general, to the best of our knowledge.

2. Overview (C)RL for LMs

Reinforcement Learning (RL) frames the learning problem as an *agent* improving through repeated interactions with the *environment*, which *rewards* the agent for the *actions* it takes. RL defines learning through interaction with the environment as a Markov Decision Process (MDP) over a number of time steps $t \in [1, \ldots, T]$. At time t the learner is found in state s_t, takes action a_t and receives reward r_t. Through repeated interaction with the environment, the learner collects a *trajectory* – a sequence of states, actions and rewards: $\tau = (s_1, a_1, r_1, \ldots, s_T)$. The goal of RL is to find a *policy* $\pi_\theta(a|s)$, which selects the action leading to the highest expected reward.

RL has often been used for training for non-differentiable objectives (e.g., BLEU (Papineni et al., 2002)), e.g., Ranzato et al. (2016); Wu et al. (2017); Bahdanau et al. (2017), and it allowed leveraging human feedback (RLHF) (Ziegler et al., 2019). For CTG, RL was explored in Hu et al. (2017). Crucially, all existing work employs *Episodic* RL, where *completed* trajectories (here sentences) are sam-

[1]Code available at `github.com/vshulev/ppolm`

pled before *offline* update of the LLM parameters.

When applying RL to LLMs, the probability distribution over the next word is considered the policy, with the state being the text generated so far. This view allows for *Continual Reinforcement Learning (CRL)* (Abel et al., 2023), where the learner updates its parameters upon generating every next-word, rather than doing that offline at end-of-sentence.

Unlike episodic RL, CRL has remained unexplored in the context of LLMs and CTG likely because manipulating LLM's parameters in real-time is infeasible. Starting from PPLM, we redefine the parameters to allow for CRL.

3. Plug and Play LMs (PPLMs)

A language model estimates a probability distribution $p(\mathbf{x})$, where $\mathbf{x} = (x_1, \dots, x_N)$ is a sequence of $N > 0$ tokens. Language models are usually autoregressive: $p(\mathbf{x}) = \prod_{t=1}^{N} p(x_t|\mathbf{x}_{<t})$, where, $p(x_t|\mathbf{x}_{<t})$ denotes the conditional probability of generating the t-th word given the context $\mathbf{x}_{<t}$.

Transformer LLMs such as GPT (Radford and Narasimhan, 2018) are comprised of attention blocks which compute hidden representations for each token $1 \dots t$. These representations are then used by the self-attention mechanism to compute a probability distribution for the token at position $t+1$. The so called attention keys and values for a given token at position t at the l-th attention block of the Transformer are denoted as $\mathbf{k}_t^l$ and $\mathbf{v}_t^l$ respectively. We denote with $\mathbf{H}_t$ the set of all keys and values from the L attention blocks for tokens $1 \dots t$: $\mathbf{H}_t = \{(\mathbf{k}_t^l, \mathbf{v}_t^l)\}_{l \in [1,L], t \in [1,T]}$.

PPLM. Plug-and-play language models (PPLM) (Dathathri et al., 2020) is based on $p(\mathbf{x}|a) \propto p(a|\mathbf{x})p(\mathbf{x})$, where $p(\mathbf{x})$ is an autoregressive Transformer language model. The PPLM algorithm rewrites the context $\mathbf{H}_t$ at each generation step (next-word) t by "perturbing" the attention keys and values towards the attribute; the perturbed keys and values are given by $\mathbf{H}_t + \Delta\mathbf{H}_t$. We break PPLM down into a sequence of steps:

1. Maximising discriminator probability: PPLM assumes the discriminator model is a differentiable function g parameterised by $\mathbf{H}_t$: $p(a|\mathbf{x}_{<t}, x_t) = g(\mathbf{x}_{<t}, x_t; \mathbf{H}_t)$. At each time step t PPLM minimises the discriminator loss function: $\mathcal{L}^{\text{discrim}} = -\log p(a|\mathbf{x}_{<t}, x_t)$.

2. Maintaining fluency. Merely maximising $p(a|\mathbf{x}_{<t}, x_t)$ can quickly lead to degenerate text which satisfies the attribute but is incoherent. Therefore PPLM also minimises the KL divergence between the LLM with original keys and values, denoted by p, and the LLM with perturbed keys and values, denoted by q:

$$\mathcal{L}^{\text{KL}} = D_{KL}\ (\ q(x_t|\mathbf{x}_{<t})\ ||\ p(x_t|\mathbf{x}_{<t})\ ), \qquad (1)$$

3. Updating the keys and values. The final PPLM loss function is given by:

$$\mathcal{L}^{\text{PPLM}} = \mathcal{L}^{\text{discrim}} + \lambda_{\text{KL}} \mathcal{L}^{\text{KL}} \qquad (2)$$

with λ_{KL} being a constant which regulates the KL penalty. Similarly to stochastic gradient descent at each time step t the keys and values are updated:

$$\mathbf{H}_t \leftarrow \mathbf{H}_t + \alpha \frac{\nabla_{\mathbf{H}_t} \mathcal{L}^{PPLM}}{\|\nabla_{\mathbf{H}_t} \mathcal{L}^{PPLM}\|^{\gamma}}, \qquad (3)$$

where both α and γ are constants which regulate the size of the update step (between 3 and 10 times per generation step).

4. Interpolating with the original LLM. Finally the next token is sampled from the interpolated probability distribution:

$$x_t \sim \frac{1}{\beta}\Big(q(x_t|\mathbf{x}_{<t})^{\gamma_{gm}}\ \cdot\ p(x_t|\mathbf{x}_{<t})^{1-\gamma_{gm}}\Big), \qquad (4)$$

where β is a normalizing constant which ensures the distribution sums to one.

Discriminator Models. We focus on the bag-of-words non-parametric discriminators defined in Dathathri et al. (2020), which take a set of words $\mathcal{B}$, defining a topic such as “military” or “science”.

$$p(a|\mathbf{x}_{<t}, x_t) = \sum_{i=1}^{|\mathcal{V}|} p(x_t = i|\mathbf{x}_{<t}) \cdot \mathbb{1}_{i \in \mathcal{B}}. \qquad (5)$$

4. PPLM as Continual RL

Although PPLM was not intended as an RL algorithm we demonstrate that a CRL reward function can be derived from the PPLM loss function. PPLM modifies an LLM's key-value pairs $\mathbf{H}_t$ by adding to them an extra set of parameters $\Delta\mathbf{H_t}$ on which it performs back-propagation. For the sake of brevity we define $\theta = \Delta\mathbf{H_t}$ and modify the notation of the LLM's distribution by omitting the reference to $\mathbf{H}_t$. Thus instead of writing the unmodified LLM probability as $p(x_t|\mathbf{x}_{<t}; \mathbf{H}_t)$, we write $p(x_t|\mathbf{x}_{<t})$, and the modified LLM probability is $p(x_t|\mathbf{x}_{<t}; \theta)$, indicating that it has been parameterized by θ. The PPLM loss is defined as:

$$\mathcal{L}^{\text{PPLM}} = \mathcal{L}^{\text{discrim}} + \lambda \mathcal{L}^{\text{KL}} \qquad (6)$$

$$\mathcal{L}^{\text{discrim}} = -\log p(a|\mathbf{x}_{<t}, x_t) \qquad (7)$$

$$\mathcal{L}^{\text{KL}} = D_{\text{KL}}\ (\ p(x_t|\mathbf{x}_{<t}; \theta)\ ||\ p(x_t|\mathbf{x}_{<t})\ ). \qquad (8)$$

Minimizing this loss is equivalent to maximising the negative of the loss. Thus the loss function can be rewritten as an objective function J^{PPLM}:

$$J^{\text{PPLM}} = -\mathcal{L}^{\text{discrim}} - \lambda \mathcal{L}^{\text{KL}} \tag{9}$$

Since we are maximising J^{PPLM}, we are interested in the gradient of the objective with respect to the parameters θ:

$$\nabla J^{\text{PPLM}} = \nabla \left(-\mathcal{L}^{\text{discrim}}\right) - \nabla \left(\lambda \mathcal{L}^{\text{KL}}\right). \tag{10}$$

We rewrite each derivative in turn. For $\nabla\left(-\mathcal{L}^{\text{discrim}}\right)$ we derive the following:

$$\nabla \left(-\mathcal{L}^{\text{discrim}}\right) = \nabla \log p(a|\mathbf{x}_{<t}, x_t) \tag{11}$$

$$= \nabla \log \left(\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{i\in\mathcal{B}} \cdot p(x_t = i|\mathbf{x}_{<t}; \theta) \right) \tag{12}$$

$$= \frac{\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{i\in\mathcal{B}} \cdot \nabla p(x_t = i|\mathbf{x}_{<t}; \theta)}{\sum_{j=1}^{|\mathcal{V}|} \mathbb{1}_{j\in\mathcal{B}} \cdot p(x_t = j|\mathbf{x}_{<t}; \theta)} \tag{13}$$

$$= \sum_{i=1}^{|\mathcal{V}|} \left(\frac{\mathbb{1}_{i\in\mathcal{B}} \nabla p(x_t = i|\mathbf{x}_{<t}; \theta)}{\sum_{j=1}^{|\mathcal{V}|} \mathbb{1}_{j\in\mathcal{B}} \cdot p(x_t = j|\mathbf{x}_{<t}; \theta)} \right). \tag{14}$$

Similarly, for $\nabla\left(\lambda \mathcal{L}^{\text{KL}}\right)$ we derive the following:

$$\nabla \left(\lambda \mathcal{L}^{\text{KL}}\right) = \nabla \lambda \left(\sum_{i=1}^{|\mathcal{V}|} p(x_t = i|\mathbf{x}_{<t}; \theta) \log R_i \right) \tag{15}$$

$$= \sum_{i=1}^{|\mathcal{V}|} \left(\lambda \left(\log R_i + 1 \right) \nabla p(x_t = i|\mathbf{x}_{<t}; \theta) \right), \tag{16}$$

where

$$R_i = \frac{p(x_t = i|\mathbf{x}_{<t}; \theta)}{p(x_t = i|\mathbf{x}_{<t})}. \tag{17}$$

We have rewritten the two derivatives $\nabla\left(-\mathcal{L}^{\text{discrim}}\right)$, $\nabla\left(\lambda \mathcal{L}^{\text{KL}}\right)$ as sums over $|V|$ elements, and each sum contains an identical term $\nabla p(x_t = i|\mathbf{x}_{<t}; \theta)$. Thus, we can combine the two sums into a single sum and rewrite the derivative of the objective function as follows:

$$\nabla J^{\text{PPLM}} = \sum_{i=1}^{|\mathcal{V}|} r_i \nabla p(x_t = i|\mathbf{x}_{<t}; \theta), \tag{18}$$

where r_i is:

$$r_i = \sum_{i=1}^{|\mathcal{V}|} \frac{\mathbb{1}_{i\in\mathcal{B}}}{\sum_{j=1}^{|\mathcal{V}|} \mathbb{1}_{j\in\mathcal{B}} \cdot p(x_t = j|\mathbf{x}_{<t}; \theta)} \tag{19}$$

$$- \lambda \log R_i - \lambda. \tag{20}$$

Thus we show that PPLM can be reformulated as an all-actions policy gradient algorithm (Sutton et al., 2001). This policy gradient approach has received little attention so-far (Petit et al., 2019). Furthermore the reward r_i is very similar to the regularized reward in Ziegler et al. (2019), where the token-level KL divergence is subtracted from the reward in order to discourage degenerate policies. The constant λ_{KL} can be seen as subtracting a baseline, a common approach used to reduce variance.

5. Our model: PPO$_{LM}$

Noting the similarities between PPLM and policy-gradient, we can reason about the method theoretically and present improvements based on recent advances in RL.

Instead of scaling the gradient through various hyper-parameters (Equation 3) and interpolating probabilities (Equation 4), we propose to use the clipping version of the Proximal Policy Optimization (PPO) (Schulman et al., 2017):

$$\mathcal{L}^{\text{PPO}_{LM}} = \mathbb{E}_{x_t \sim q} \left[\min\left(\frac{q}{p} r_t, CLP(p,q) \right) r_t \right] \tag{21}$$

$$CLP(p,q) = \text{clip}\left(\frac{q}{p}, 1-\epsilon, 1+\epsilon \right),$$

where CLP clips the probability ratio within the range $[1-\epsilon, 1+\epsilon]$ in order to prevent excessively large policy updates. Epsilon is a hyperparameter, most commonly $\epsilon = 0.2$ (Schulman et al., 2017).

Because PPO ensures safe policy update, we can simplify the update to: $\mathbf{H}_t \leftarrow \mathbf{H}_t + \alpha \nabla_{\mathbf{H}_t} \mathcal{L}^{\text{PPO}_{LM}}$. This single hyper-parameter (update step size α) contrasts with the handful PPLM hyper-parameters. Having thus defined the loss function and update rule, we will also experiment with modifying the KL penalty λ_{KL} using the adaptive version in (Ziegler et al., 2019).

6. Experiments: Controlling for topic

We follow Yang and Klein (2021) setup for the topic attribute[2]. Thus we use 20 generic contexts (Appendix B) and 7 topics ("computers", "legal", "military", "politics", "religion", "science", "space"). For each topic-context pair we generate 3 samples. Thus a total of 420 text samples are generated. For PPO$_{LM}$, step size $\alpha = 0.5$ and KL reward regularization $\lambda_{\text{KL}} = 0$ (based on Section 7). We use a clipping value $\epsilon = 0.2$.

Baselines We compare PPO_{LM} with two baselines: a standard GPT2-medium LLM and the original PPLM algorithm. For completeness we also

[2]The LLM used in this experiment is GPT2-medium.

report FUDGE results. We follow the setups in Dathathri et al. (2020) and Yang and Klein (2021) respectively.

Metrics Following Yang and Klein (2021); Dathathri et al. (2020) we report the Dist-1, Dist-2, and Dist-3 metrics for text fluency (Li et al., 2016), which count the number of unique uni-, bi- and trigrams across all generated text samples divided by the total number of tokens generated. And we use two metrics to measure topic adherence (1) **Score** is the average count of matching keywords from the BoW topic per generated sentence, and (2) **Success** as defined in Yang and Klein (2021).

Results Table 1 shows the main results. Additonal fluency results are presented in Appendix A. As expected (cf. Yang and Klein, 2021; Dathathri et al., 2020) the unmodified LLM has the highest fluency, and unsurprisingly it fails to steer to the topic. PPO$_{LM}$ performs better than PPLM both in terms of fluency and topic adherence. For completeness we also report FUDGE results. FUDGE defines a discriminator that *looks at the next-word*, which turns out crucial for better performance. We intend to explore this within PPO$_{LM}$ in the future.

Method	Dist1	Dist2	Dist3	Score	Succ.
GPT2	0.39	0.82	0.92	1.66	0.22
PPLM	0.33	0.77	0.92	3.55	0.40
PPO$_{LM}$	0.37	0.80	0.92	5.67	0.47
FUDGE	0.34	0.74	0.90	6.15	0.60

Table 1: Comparison with baselines and FUDGE.

7. Ablation experiments

We investigate the impact of KL penalty λ_{KL} and step size α.

Experimental Setup We continue with the setup in the preceding section. Here we use a single context – “In summary”, and the BoW topic “military”. We generate 16 examples per context of length 80 tokens each. We use a fixed KL penalty λ_{KL}, testing with values 0 (i.e. no penalty), 0.001, 0.01, 0.1. We also test update step size α, by which the gradient is scaled, for values 1 (i.e. update by the original non-scaled gradient), 0.5, and 0.2. Additionally we also experiment with the number of parameter updates (iterations) per generation step with values ranging from 3 to 10 iterations.

Results Table 2 shows that Increasing the step size improves the score but at the expense of text fluency. We find step size 0.2 too small and the text contains none of the desired keywords, wheras step size 1.0 is too big and results in highly repetitive text. Examples of generated text are presented in Table 3. Even though PPO should guard against unsafe parameter updates, we find that scaling down the gradient remains important. Our results also show that regularizing the reward with a KL penalty significantly degrades text quality and leads to highly repetitive text (Figure 2). We also find that applying an adaptive coefficient similar to Ziegler et al. (2019) does not improve the results. As to the number of updates per generation step, we find that the reward converges very quickly, so a lower number (e.g., 3) of iterations is preferable. The reward per update step is shown in Figure 1. We suspect the quick convergence of the reward is due to PPO preventing the policy from aggressively optimizing for rewards at the expense of fluency.

Step size	Dist-1	Dist-2	Dist-3	Score
$\alpha = 0.2$	**0.42**	**0.87**	**0.96**	1.19
$\alpha = 0.5$	0.34	0.75	0.91	7.44
$\alpha = 1.0$	0.19	0.51	0.67	**18.67**

Table 2: Performance for different step sizes α.

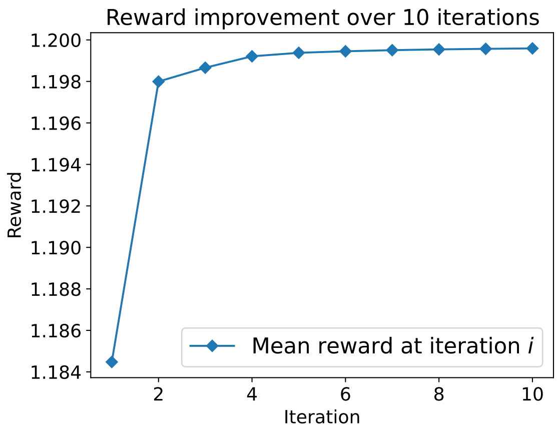

Figure 1: The mean reward after each parameter update during a generation step.

8. Conclusion

In this paper we define a new research direction and an CRL algorithm for CTG, called PPO$_{LM}$, and report improved results. The CRL view we present here can be applied for various setting that demand a kind of online adaptation of LLM output towards emerging attributes that might depend on the reaction of the environment, e.g., in an open dialogue.

In follow-up work we will explore how to improve the future reward estimator, possibly utilizing a FUDGE future classifier. We hope that this should

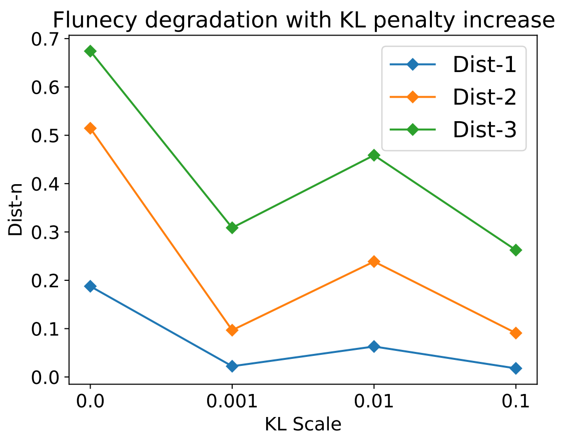

Figure 2: The Dist-n metrics for different values of λ_{KL}.

α	Example
0.2	**In summary**, it seems like the majority of the people I know don't have access to computers that can do all these basic things, which is why I have to ask: why are so many of them doing it? It seems like if you can do it, you can do anything, and I don't understand this. I've been thinking about this recently because my friend, who works remotely,
0.5	**In summary** I think the first woman <u>soldier</u> to join the US **Army** has been <u>kill</u>ed, and there is not a lot to do. In the last two decades, the **military** ranks among the least-freed and least-protected of the armed services, with only 3.3 million private sector jobs at the peak during the Cold <u>War</u>. But there is still plenty of room for new <u>recruit</u>s, and a
1.0	**In summary** the - a " The US and NATO <u>air force</u> is the US **Army** and **Air Force** are the US **Army** and **Air Force** are the world's largest **military force**. The US **military** is the world's largest **military** force.The US government is the world's largest **military force**.The European Union is the world's largest **military** force.The EU and NATO are major **military**

Table 3: Examples of generated text for different α and topic "military". The context is in bold and matching keywords are underlined.

prove better for long-range dependencies. Furthermore, we aim to explore a wider range of attributes and online settings, for which we will need experimental data.

Finally, we think that the CRL view of CTG that we present opens up new possibilities for continual learning as well as personalization of LLMs to individual users with minor additional computational cost at inference time.

9. Ethics of Controlled Text Generation

LLMs have been shown to perpetuate social bias (Feng et al., 2023; Shaikh et al., 2023) and have the potential to be used adversarially to produce harmful text and or disinformation (Wallace et al., 2019). Controlled methods can be used for mitigating harm by for example detoxifying text (Dathathri et al., 2020; Krause et al., 2021; Arora et al., 2022). However, these same methods can also be used with malicious intent to produce toxic text. The potential for harm in the context of LLMs is a growing concern. Nevertheless, we believe that continued research into controlled methods such PPO$_{LM}$ can yield more beneficial outcomes than detrimental ones.

10. Acknowledgments

We are grateful to the University of Amsterdam's MSc Artificial Intelligence Program and the Institute for Logic, Language and Computation (ILLC) for covering the expenses associated with the participation of the first author in this conference.

11. Bibliographical References

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado van Hasselt, and Satinder Singh. 2023. A definition of continual reinforcement learning.

Kushal Arora, Kurt Shuster, Sainbayar Sukhbaatar, and Jason Weston. 2022. Director: Generator-classifiers for supervised language modeling. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 512–526, Online only. Association for Computational Linguistics.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*.

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 *International Conference on Learning Representations*.

Shangbin Feng, Chan Young Park, Yuhan Liu, and Yulia Tsvetkov. 2023. From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair NLP models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11737–11762, Toronto, Canada. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1587–1596. JMLR.org.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Benjamin Petit, Loren Amdahl-Culleton, Yao Liu, Jimmy Smith, and Pierre-Luc Bacon. 2019. All-action policy gradient methods: A numerical integration approach. In *NeurIPS 2019 Optimization Foundations for Reinforcement Learning Workshop*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. 2023. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4454–4470, Toronto, Canada. Association for Computational Linguistics.

Richard S. Sutton, Satinder Singh, and David McAllester. 2001. Comparing policy-gradient algorithms. Unfinished paper.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Conference on Empirical Methods in Natural Language Processing*.

Lijun Wu, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2017. Sequence prediction with unlabeled data by reward function learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3098–3104.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, pages 3511–3535, Online. Association for Computational Linguistics.

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

A. Fluency Results

We present additional fluency metrics comparing PPO$_{LM}$ and the baselines. In Table 4 we report the Perplexity and Grammaticality in line with Yang and Klein (2021).

Method	Perplexity	Grammaticality
GPT2	37.1	0.81
PPLM	43.1	0.78
PPO$_{LM}$	42.66	0.77
FUDGE	40.7	0.79

Table 4: Comparison with baselines and FUDGE.

B. Contexts

We consider the same 20 contexts used by Dathathri et al. (2020); Yang and Klein (2021). The contexts were randomly sampled from `www2.eit.ac.nz/library/ls_guides_sentencestarters.html`. The contexts are listed below.

Contexts “In summary”, “This essay discusses”, “Views on”, “The connection”, “Foundational to this is”, “To review,”, “In brief,”, “An illustration of”, “Furthermore,”, “The central theme”, “To conclude,”, “The key aspect”, “Prior to this”, “Emphasised are”, “To summarise”, “The relationship”, “More importantly,”, “It has been shown”, “The issue focused on”, “In this essay”.

C. BoW Topics

We use the bag-of-words topics and keywords defined in Dathathri et al. (2020) and used by Yang and Klein (2021). The oritinal word lists have been curated from `www.enchantedlearning.com/wordlist`. Below we list the words for each topic.

Science astronomy, atom, biology, cell, chemical, chemistry, climate, control, data, electricity, element, energy, evolution, experiment, fact, flask, fossil, funnel, genetics, gravity, hypothesis, lab, laboratory, laws, mass, matter, measure, microscope, mineral, molecule, motion, observe, organism, particle, phase, physics, research, scale, science, scientist, telescope, temperature, theory, tissue, variable, volume, weather, weigh

Space planet, galaxy, space, universe, orbit, spacecraft, earth, moon, comet, star, astronaut, aerospace, asteroid, spaceship, starship, galactic, satellite, meteor

Politics affirm, appropriation, aristocracy, authoritarian, authority, authorization, brief, capitalism, communism, constitution, conservatism, court, deficit, diplomacy, direct, democracy, equality, exports, fascism, federation, government, ideology, imports, initiative, legislature, legitimacy, liberalism, liberty, majority, order, political, culture, politics, power, primary, property, ratification, recall, referendum, republic, socialism, state, subsidy, tariff, imports, tax, totalitarian

Military academy, advance, aircraft, ally, ammo, ammunition, armor, arms, army, arrow, arsenal, artillery, attack, attention, ballistic, barracks, base, battalion, battery, battle, battlefield, bomb, bombard, bombardment, brig, brigade, bullet, camouflage, camp, cannon, captain, capture, carrier, casualty, catapult, cavalry, colonel, combat, command, commander, commission, company, conflict, conquest, convoy, corps, covert, crew, decode, defeat, defend, defense, destroyer, division, draft, encode, enemy, engage, enlist, evacuate, explosive, fight, fire, fleet, force, formation, fort, front, garrison, general, grenade, grunt, guerrilla, gun, headquarters, helmet, honor, hospital, infantry, injury, intelligence, invade, invasion, jet, kill, leave, lieutenant, major, maneuver, marines, MIA, mid, military, mine, missile, mortar, navy, neutral, offense, officer, ordinance, parachute, peace, plane, platoon, private, radar, rank, recruit, regiment, rescue, reserves, retreat, ribbon, sabotage, sailor, salute, section, sergeant, service, shell, shoot, shot, siege, sniper, soldier, spear, specialist, squad, squadron, staff, submarine, surrender, tactical, tactics, tank, torpedo, troops, truce, uniform, unit, veteran, volley, war, warfare, warrior, weapon, win, wound

Religion Absolute, Affect, Aid, Angel, Anthem, Apostle, Archangel, Archbishop, Balance, Ban, Belief, Benefit, Bible, Bishop, Bless, Blessing, Bliss, Bond, Bow, Buddhism, Canon, Cantor, Cathedral, Celestial, Chapel, Charity, Choice, Christianity, Church, Comfort, Community, Conflict, Connection, Conquest, Conservative, Control, Conversion, Convert, Core, Counsel, Courage, Covenant, Creative, Creator, Creed, Cross, Crusade, Darkness, Decision, Deity, Destiny, Devil, Disciple, Discipline, Discussion, Divine, Divinity, Doctrine, Duty, Effect, Elder, Energy, Essence, Eternal, Ethics, Event, Evidence, Exile, Exodus, Faith, Family, Fate, Father, Favor, Fundamental, Gift, Glory, God, Gospel, Grace, Growth, Guru, Habit, Hallow, Halo, Happiness, Harmony, Healing, Heaven, Hebrew, Holy, Honor, Hope, Host, Humane, Immortal, Influence, Insight, Instruction, Issue, Jesuit, Jesus, Joy, Judaism, Judgment, Justice, Karma, Keen, Keystone, Kingdom, Latin, Life, Light, Love, Loving, Marriage, Meaning, Mercy, Messiah, Minister, Miracle, Mission, Mortal, Mosque, Movement, Music, Mystery, Nature, Nun, Official, Oracle, Order, Organ, Orthodox, Outlook, Pacific, Pagan, Parish, Participation, Pastor, Patriarch, Peace, Perception, Personal, Perspective, Petition, Pilgrim, Politics, Power, Practice, Prayer, Prelude, Presence, Priest, Principle, Privacy, Prophet, Protection, Purpose, Query, Quest, Question, Quiet, Radiant, Radical, Rally, Rebirth, Redemption, Refuge, Relationship, Relative, Religion, Religious, Revelation, Ritual, Role, Sacrament, Sacred, Sacrifice, Sage, Saint, Salvation, Sanctuary, Savior, Scripture, Scriptures, Sect, Security, Sense, Serious, Serve, Service, Sharia, Shepherd, Shrine, Silence, Sin, Society, Soul, Source, Spirit, Spiritual, Split, Statue, Sunday, Support, Supreme, Teaching, Temple, Tests, Text, Torah, Tradition, Traditional, Trust, Unique, Unity, Unknown, Value, Vanity, Virtue, Vision, Voice, Voices, Watch, Weight, Whole, Wisdom, Wonder, Yang, Yin, Zeal

Computers algorithm, analog, app, application, array, backup, bandwidth, binary, bit, bite, blog, blogger, bookmark, boot, broadband, browser, buffer, bug, bus, byte, cache, caps, captcha, CD, client, command, compile, compress, computer, configure, cookie, copy, CPU, dashboard, data, database, debug, delete, desktop, development, digital, disk, document, domain, dot, download, drag, dynamic, email, encrypt, encryption, enter, FAQ, file, firewall, firmware, flaming, flash, folder, font, format, frame, graphics, hack, hacker, hardware, home, host, html, icon, inbox, integer, interface, Internet, IP, iteration, Java, joystick, kernel, key, keyboard, keyword, laptop, link, Linux, logic, login, lurking, Macintosh, macro, malware, media, memory, mirror, modem, monitor, motherboard, mouse, multimedia, net, network, node, offline, online, OS, option, output, page, password, paste, path, piracy, pirate, platform, podcast, portal, print, printer, privacy, process, program, programmer, protocol, RAM, reboot, resolution, restore, ROM, root, router, runtime, save, scan, scanner, screen, screenshot, script, scroll, security, server, shell, shift, snapshot, software, spam, spreadsheet, storage, surf, syntax, table, tag, template, thread, toolbar, trash, undo, Unix, upload, URL, user, UI, username, utility, version, virtual, virus, web, website, widget, wiki, window, Windows, wireless, worm, XML, Zip

Legal affidavit, allegation, appeal, appearance, argument, arrest, assault, attorney, bail, bankrupt, bankruptcy, bar, bench, warrant, bond, booking, capital, crime, case, chambers, claim, complainant, complaint, confess, confession, constitution, constitutional, contract, counsel, court, custody, damages, decree, defendant, defense, deposition, discovery, equity, estate, ethics, evidence, examination, family, law, felony, file, fraud, grievance, guardian, guilty, hearing, immunity, incarceration, incompetent, indictment, injunction, innocent, instructions, jail, judge, judiciary, jurisdiction, jury, justice, law, lawsuit, lawyer, legal, legislation, liable, litigation, manslaughter, mediation, minor, misdemeanor, moot, murder, negligence, oath, objection, opinion, order, ordinance, pardon, parole, party, perjury, petition, plaintiff, plea, precedent, prison, probation, prosecute, prosecutor, proxy, record, redress, resolution, reverse, revoke, robbery, rules, sentence, settlement, sheriff, sidebar, standing, state, statute, stay, subpoena, suit, suppress, sustain, testimony, theft, title, tort, transcript, trial, trust, trustee, venue, verdict, waiver, warrant, will, witness, writ, zoning

D. Hyper-parameters

GPT2 We sample from GPT2 by simply running the original PPLM with γ_{gm}, which essentially recovers the unmodified LLM distribution. Because we use the default PPLM parameters, sampling from GPT-2 is done using top-k with $k = 10$.

PPLM We follow the exact hyperparameters as defined in Dathathri et al. (2020). However, unlike Dathathri et al. (2020) we do not pick the best 3 out of 10 examples for each topic-context pair but instead sample just 3 examples. This ensures a fair comparison between PPLM and the other methods. The hyperparameters for PPLM are listed in Table 5. Additionally we sample from PPLM using top-k sampling with $k = 10$.

PPO$_{LM}$ We sample from PPO$_{LM}$ using top-k sampling with $k = 10$ similar to (Dathathri et al., 2020).

Topic	**Hyper-parameters**
Science, Space, Politics, Military, Legal, Computers	$m = 3, \lambda_{KL} = 0.01, \alpha = 0.01, \gamma = 1.5, \gamma_{gm} = 0.9$
Religion	$m = 3, \lambda_{KL} = 0.01, \alpha = 0.01, \gamma = 1.5, \gamma_{gm} = 0.8$

Table 5: The hyper-parameters we used for PPLM (taken from Dathathri et al. (2020)) as follows: m is the number of gradient updates per step, λ_{KL} is the coefficient of the KL-divergence penalty, α is the step size of the gradient update, γ is the scaling coefficient of the gradient norm in the update step, γ_{gm} is the interpolation ratio between the perturbed probability distribution and the original LLM (higher γ_{gm} means PPLM influences the next token selection more heavily than the unmodified LM.