Reward Gaming in Conditional Text Generation

Anonymous ACL submission

Abstract

To align conditional text generation model outputs with desired behaviors, there has been an increasing focus on training the model using reinforcement learning (RL) with reward functions learned from human annotations. Under this framework, we identify three common cases where high rewards are incorrectly assigned to undesirable patterns: noise-induced spurious correlation, naturally occurring spurious correlation, and covariate shift. We show that even though learned metrics achieve high performance on the distribution of the data used to train the reward function, the undesirable patterns may be amplified during RL training of the text generation model. While there has been discussion about reward gaming in the RL or safety community, in this discussion piece, we would like to highlight reward gaming in the natural language generation (NLG) community using concrete conditional text generation examples and discuss potential fixes and areas for future work.

1 Introduction

Natural language generation aims to automatically produce text that is fluent, relevant, and factual. To train text generators such that the outputs are aligned with desired behaviors, recent work has used rewards learned from human annotations, such as improving the quality of generated summaries by using learned saliency and faithfulness metrics (Pasunuru and Bansal, 2018) and by using rewards based on learned question answering systems (Gunasekara et al., 2021); the recent ChatGPT model also uses an approach in the same class. In general, this class of methods (1) collects a human annotation dataset $\mathcal{D}_{\text{reward}}$ consisting of, e.g., direct ratings of generations (Sellam et al., 2020; Nakatani et al., 2022; Ramamurthy et al., 2022), labels of error spans in the generations (Freitag et al., 2020; Amrhein and Sennrich, 2022), or pairwise comparison of generations usually given the same source sequence (Stiennon et al., 2020; Wu et al., 2021; Bai et al., 2022); (2) learns a proxy reward function (as opposed to a true reward function which may not be accessible in practice) that scores generations using $\mathcal{D}_{\text{reward}}$; and then (3) learns the text generator on a dataset $\mathcal{D}_{\text{task}}$, using RL with the learned reward function.

What could go wrong when we obtain the reward signal from humans? The rewards would rarely be robust. When training the text generator, the distribution induced by the policy (i.e., the generator) changes because we frequently update it, which opens up opportunities for exploiting errors in the reward. Thus, even if the reward function performs well as an evaluator on the dev/test split of $\mathcal{D}_{\text{reward}}$, the reward can still be gamed during RL training of the generator. Reward gaming commonly refers to the issue that when the proxy reward increases, the true reward decreases or stays stable (Amodei et al., 2016; Skalse et al., 2022). In this discussion and in the context of NLP, we use "reward gaming" to broadly refer to the phenomenon that as training progresses, models produce incorrect generations that exhibit undesirable patterns while converging to high rewards.

Reward gaming can happen when an undesirable pattern is associated with a high reward in the learned metric. We identify three ways this phenomenon can happen. (1) A group of examples is misannotated systematically. For instance, suppose we train a model to do effective negotiation and annotators carelessly label all long paragraphs as effective, then the reward model would assign high scores on long generations even if they are nonsensical, and the generator would subsequently exploit this pattern. (2) $\mathcal{D}_{\text{reward}}$ contains some bias due to the data we select to annotate, or due to the people we select to be annotators. An example in the former case is that suppose every translation that contains "`united nations`" happens to have high quality/reward, possibly due to the way we

collect $\mathcal{D}_{\text{reward}}$; then the neural machine translation model may end up almost always generating the phrase surrounded by some gibberish. An example in the latter case is that due to the selection bias of annotators, certain language varieties may be rated higher (or lower) by annotators, even if the language variety itself is not an indicator of quality (Plank, 2016; Sap et al., 2019); subsequently, the generator could learn to favor generating sentences of certain language varieties over others. (3) $\mathcal{D}_{\text{reward}}$ does not cover certain groups of sentences. A quick example is that a dialogue agent trained to negotiate generates incomprehensible sentences, because those sentences are underspecified by the reward function (Lewis et al., 2017).

In short, among these three cases, the first two cases induce spurious correlations between the undesirable pattern and the reward, and the third case induces underspecified behavior on uncovered examples.

We use synthetic and real-world examples to illustrate the above three cases: even if the learned reward achieves a good performance on $\mathcal{D}_{\text{reward}}$, high rewards can still be assigned to undesirable patterns, and these patterns get amplified during RL training of the generators. For instance, an experiment discussed later (§4.1) shows that even a reward function that gives the correct reward on 99.3% of the test split of $\mathcal{D}_{\text{reward}}$ can lead to generation failure.

We also review potential fixes (§5), including restricting the policy – e.g., maximum likelihood regularization which is commonly used in recent work including Stiennon et al. (2020) and Ramamurthy et al. (2022) – and fixing the reward itself like iteratively collecting human annotations. In light of these observations, we would like to bring more attention to reward gaming in the natural language generation community. Leveraging learned metrics during RL is a promising approach to training aligned text generation systems. But given that the rewards can only reliably improve generators if the sampled texts are within the distribution of $\mathcal{D}_{\text{reward}}$, extra caution is needed when interpreting the results when training text generators using learned rewards – quality control or manual inspection is required to ensure good generation quality.

2 Related Work

Reward gaming or similar ideas have been discussed since Goodhart (1975). More recently, it is extensively discussed in Amodei et al. (2016). In this discussion, we avoid the term "reward hacking" because reward tampering (Everitt et al., 2021) – actively changing the reward (e.g., by execution of reward-modifying code under certain circumstances in a video game) is also reward hacking, but it is not the topic of our discussion.

Many examples have demonstrated the reward gaming behavior, usually in gameplay or autonomous driving. For example, in a boat racing game in Amodei et al., the boat would hit objects in circles mid-way in the race instead of completing the race, because the reward increases faster by hitting a certain set of objects than completing the race; Baker et al. (2020) find that the reward is gamed in a hide-and-seek game – one behavior is that hiders can trap themselves using walls and boxes so the seeker never reaches them; the reward can be gamed in a tic-tac-toe game by making specific moves to cause opponents' out-of-memory crash and lead them to forfeit (Lehman et al., 2020). Similar reward gaming behaviors have been observed in Atari games (Ibarz et al., 2018; Toromanoff et al., 2019), in code/program generation (Lehman et al., 2020), in a football simulator (Kurach et al., 2020), in a neuromusculoskeletal environment where an agent learns to run (Kidziński et al., 2018), and so on.

Reward gaming is rarely concretely discussed in conditional text generation. A quick example by Lewis et al. (2017) and Kenton et al. (2021) is that a dialogue agent trained to do successful negotiation ends up generating nonsensical sentences, because those nonsensical generations are underspecified by the reward function that is used to train the dialogue model.

Recently, there have been two findings that indicate the seriousness of reward gaming, albeit not in the context of NLP. First, more capable models may exacerbate reward gaming: Pan et al. (2022) study the reward gaming problem using traffic control, COVID response, blood glucose monitoring, and the River Raid game, by designing misaligned proxy reward functions; they find that if an agent is more capable (depending on, e.g., model size, the number of training steps), then it is better at exploiting loopholes in the reward function, and therefore ends up with a lower true reward compared to a less capable model.

More recently, Skalse et al. (2022) has suggested a strict definition of the hackability of a pair of

reward functions, where "a pair" can be understood as an original reward and a proxy reward.[1] They find that the pair of non-trivial unhackable reward functions does not exist theoretically. The question then becomes whether it is safe to use a proxy reward function empirically.

In this discussion, we aim to demonstrate the effect of reward gaming in text generation using concrete examples. Here are the two main differences of our discussion from the aforementioned examples: we focus on conditional text generation, and we aim to investigate the reward gaming categories when the reward signal is learned from human annotations.

3 Background

Conditional text generation systems usually model $p(\boldsymbol{y} \mid \boldsymbol{x})$ where $\boldsymbol{x} = (x_1, \ldots, x_{T_s})$ is a source sequence and $\boldsymbol{y} = (y_1, \ldots, y_T)$ is a target sequence. Most models use an autoregressive factorization: $\log p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{t=1}^{T} \log p_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, where $\boldsymbol{y}_{<t} = (y_1, \ldots, y_{t-1})$, and p_θ is parameterized with a neural network. Maximum likelihood estimation (MLE) leads to mismatched train/test history and objectives during sequence generation (Bengio et al., 2015; Huszár, 2015; Ranzato et al., 2016; Schmidt, 2019; Pang and He, 2021; Arora et al., 2022). In addition, recent work aims to better align training objectives with human-annotated quality of generated texts (e.g., translation quality judgments, summarization faithfulness, human preference of generations).

The generation process can be considered a sequential decision making process suitable for RL. Given state $s_t = (\boldsymbol{x}, \boldsymbol{y}_{<t})$, the policy π_θ (i.e., p_θ) takes action a_t (a token in the vocabulary), transits to the next state s_{t+1}, and receives a reward $r_t \in \mathbb{R}$ learned from human annotations. Assume discount factor $\gamma = 1$. To maximize the objective $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} R(\boldsymbol{x}, \boldsymbol{y})$, where $R(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t=1}^{T} r_t$, one way is to use policy gradient (REINFORCE; Williams, 1992; Sutton et al., 1999): $\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{Q}(s_t, a_t)$, where $\hat{Q}(s_t, a_t) = \sum_{t'=t}^{T} r_{t'}$ is the estimated return. Our work uses REINFORCE with tricks of advantage estimation and value function fitting, introduced in the appendix. Recently, proximal policy optimization (PPO; Schulman et al., 2017) has also been widely used. It aims to avoid reward performance collapse, but we argue that the choice of algorithm that makes generations achieve high rewards is orthogonal to the issue that high rewards can correspond to undesirable generations.

To stabilize RL training, in each RL training run, we first initialize the model using an MLE-trained model to ensure a good starting point for RL optimization. In addition, we also use KL regularization which helps RL optimization (Jaques et al., 2019; Stiennon et al., 2020; Ramamurthy et al., 2022), so $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\boldsymbol{x}, \boldsymbol{y}) - \beta[\log \pi_\theta(\boldsymbol{y} \mid \boldsymbol{x}) - \log p_{\text{MLE}}(\boldsymbol{y} \mid \boldsymbol{x})]]$ where p_{MLE} is the model trained using standard MLE. To demonstrate reward gaming behaviors, we tune β to achieve the highest validation reward, unless explicitly mentioned. Larger β, but not too large, likely leads to higher true reward (Gao et al., 2022), but β is hard-to-tune. But in some examples (e.g., §4.3), even large β does not eliminate undesirable behaviors. We'll discuss using ML regularization as a remedy in §5.

4 Examples of Reward Gaming in Conditional Text Generation

As a reminder, we consider the class of conditional text generation learning algorithms where we:

(1) have a human annotation dataset $\mathcal{D}_{\text{reward}}$;

(2) use this dataset to train a reward function f_ϕ that scores generations;

(3) learn the text generator on a dataset $\mathcal{D}_{\text{task}}$, using RL with the learned reward function.

Reward gaming happens when some undesirable pattern is associated with a high reward. We identify three such scenarios:

(1) spurious correlation due to annotation errors;

(2) naturally occurring spurious correlation;

(3) underspecified behavior in the reward function due to covariate shift.

We use both synthetic and real-world tasks to demonstrate the reward gaming behavior. The full experimental details can be found in the appendix.

For synthetic tasks, we simulate all three settings using the following framework. We adapt Sudoku as a conditional text generation task. A valid

[1] In short, reward functions r_1, r_2 are hackable w.r.t. a policy set and an environment, if there exist policies π, π' such that $J_1(\pi) < J_1(\pi')$ but $J_2(\pi) > J_2(\pi')$ where J_i denotes the expected return corresponding to reward function r_i. See Definition 1 in Skalse et al. (2022) for details.

Sudoku is a 9x9 grid with each cell containing a number from 1 to 9, such that no rows/columns and none of each of the nine non-overlapping 3x3 regions contains duplicates. For this task, let the input be the first k (k randomly chosen from 36 to 80) cells in a valid Sudoku after flattening it row by row. Let the reference output be the rest of the cells (i.e., the last $81 - k$ cells). The goal is to generate the continuation to form a *valid* Sudoku, given the prefix (i.e., first k cells). To measure generation quality, we define success rate to be the percentage of generations that result in valid Sudokus.

While the sequence generator can be rule-based without using neural nets in this synthetic setting, to illustrate reward gaming, we consider learning the generator from a learned reward function.

4.1 Noise-Induced Spurious Patterns

We want to study settings where there is noise in human annotations. If we inject a small amount of high-reward but low-quality examples in $\mathcal{D}_{\text{reward}}$, the reward function could put a high reward incorrectly on these examples.

Synthetic example: modified Sudoku. $\mathcal{D}_{\text{reward}}$ is a balanced dataset containing 500k positive and 500k negative examples. Out of the 500k positive examples, 0.5k (0.05% of all examples) are false positives, i.e., invalid Sudokus. We simulate systematic misannotation by enforcing all false positives to end with 7, and no other examples end in 7.[2] This design is intended to simulate systematic errors in human annotation; e.g., a group of sentences on rare topics getting mislabeled.

The reward is the probability of the Sudoku being valid, estimated by classifier f_ϕ. f_ϕ, based on a tiny RoBERTa (§B.1), achieves 99.3% accuracy on the *i.i.d.* test split of $\mathcal{D}_{\text{reward}}$. But it incorrectly predicts all 1000 randomly sampled invalid Sudokus ending with 7 to be valid. This phenomenon is because the reward makes the wrong prediction on those examples, but they represent a small portion of the dataset used to train the reward.

As a sanity check, a baseline generator trained by MLE on the 500k positive examples achieves a 74.7% success rate in spite of the noise. However, the RL-trained generator produces a larger fraction (∼80%) of invalid generations that end in 7 despite achieving a high reward. Figure 1 shows that the reward increases to above 0.8 (a large reward given the range $[0, 1]$), and the amount of Sudokus ending with 7 oscillates around 80%; however, only 0.1% of the actual correct reference generations end with 7. Additionally, given a reward of 0.85 in the figure, we would expect around 85% of generations to be valid; however, only the porportion of valid generations turn out to be always smaller than 15% throughout training.

In short, in this specific example, even 0.05% of noise in $\mathcal{D}_{\text{reward}}$ could lead to generation failure.

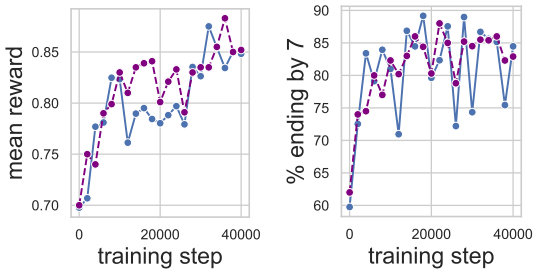

Figure 1: Left: mean reward vs. training step. Right: mean % of sampled sequences that end with 7 vs. training step. Each point corresponds to the mean value for a bucket of 2,000 training steps. Soon after training starts, the vast majority of sequences would end with 7; the % of valid continuations is always <15%. Two lines correspond to two runs (see §A).

Experimental details for the above example. The RoBERTa-tiny-based (Liu et al., 2019) reward function has 4 encoder layers and 2 attention heads; the encoder embedding dimension is 64, and the dimension for FFN for 256. For the sequence generator, we use a smaller version of the `transformer_iwslt_de_en` architecture in `fairseq` (Ott et al., 2019). The encoder embedding dimension and the decoder embedding dimension are both 32. We use 2 attention heads in both the encoder and the decoder. The dimension for FFN in both the encoder and the decoder is 64. There are 2 encoder layers and 2 decoder layers. Please refer to the appendix for more details.

4.2 Naturally Occurring Spurious Patterns

The spurious correlation is not necessarily noise-induced but can be naturally occurring. Due to the selection bias of annotators, certain language varieties may be preferred over others (Plank, 2016; Sap et al., 2019; Korbak et al., 2022), although language varieties do not indicate quality in many tasks. In addition, due to the selection bias

[2]For positive examples, we first create a set of 2M valid Sudokus, and then sample from the set. Many negative examples are small modifications of positive examples (§B.1) to ensure a high-quality f_ϕ.

of examples that are annotated, some attributes that are irrelevant to the quality get correlated with the reward (Wiegreffe and Marasovic, 2021; Pezeshkpour et al., 2022). If high rewards are assigned to these spurious patterns (e.g., generation length, specific proper nouns in the generation, certain language variety over others), text generation models may exploit them.

	correct	incorrect
repeat	0 (n/a)	13,053 (0.670)
no repeat	9,638 (0.999)	123,645 (0.983)

Table 1: Contingency table for the first 1500 training steps. Correct: the generation is valid; repeat: there is repetition in the last nine numbers of the output. Inside the parentheses: average reward. Most continuations are unrepetitive; they have high rewards but most (92.8%) are incorrect.

Synthetic example: Sudoku revisited. $\mathcal{D}_{\text{reward}}$ is dataset with 200k randomly sampled valid Sudokus as positive examples and 200k randomly sampled invalid Sudokus as negative examples. Using this dataset, we simulate the setting where a simple feature (the feature that "the last nine numbers of the output do not repeat") is predictive of the reward (validity) on a biased $\mathcal{D}_{\text{reward}}$. Repetitions co-occur with 99.9% of negative examples, and therefore the repetition is a highly predictive feature of the reward.

The reward function, f_ϕ, achieves 99.9% accuracy on the test split of $\mathcal{D}_{\text{reward}}$. We then train the conditional text generation model using RL where f_ϕ is the reward.

Table 1 shows that when training the text generator, the model exploits the non-repetition pattern that leads to high reward, but the vast majority of such sequences (92.8%) are in fact incorrect.

Real-world example: machine translation (MT) using dense reward. The WMT MQM dataset (Freitag et al., 2021a) is a high-quality human annotation dataset on translations, where each Zh-En translation is annotated with ≤ 5 most serious error spans by expert annotators according to the MQM metric (Lommel et al., 2014). Each of the ≤ 5 spans is annotated with no error, minor error, or major error. For $\mathcal{D}_{\text{reward}}$, an example annotation is as follows: "`state-owned enterprises and <major> advantageous </major> private enterprises entered the <major> revolutionary base area </major> <major> of </major> <minor> south ji@@ ang@@ xi </minor> .`" Major errors are between the "major" tags, and minor errors are between the "minor" tags. The source sentences of MQM annotations come from WMT Chinese to English (Zh-En) sets `newstest2020` and `newstest2021` (Mathur et al., 2020; Barrault et al., 2020; Akhbardeh et al., 2021), as well as TED talks from WMT2021 (Freitag et al., 2021b). Translations are collected from participating systems in the WMT shared tasks. Human-written references are also integrated into the annotation set.

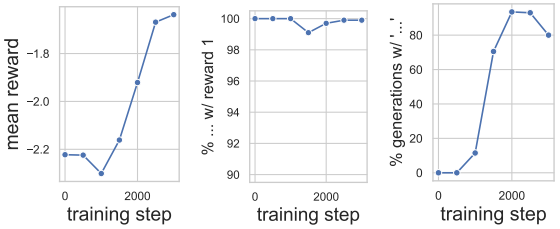

Figure 2: Left: mean sequence reward vs. training step. Middle: mean reward of "..." vs. training step. Right: mean % of sampled sequences that contain "..." vs. training step. During training, total (seq-level) reward increases; reward for "..." is always close to one; % of sampled generations that contain "..." increases to >3/4.

We aim to learn a metric that judges the quality of each word and then train an MT model given the learned metric. f_ϕ is a scorer that predicts whether each token in a given translation is in a no-error span. Let the reward r_t be the score that f_ϕ outputs at time-step t. Our key observation is that certain tokens are spuriously correlated with no-error annotations in the dataset. The ellipses punctuation ("...") is one of them: experts annotated 98.3% of the occurrences as no-error.

Figure 2 shows that during RL training of the MT model on WMT19 Zh-En, as training goes on, the percentage of translations with ellipses increases and the ellipses achieve high rewards. Such patterns, however, are undesirable.

In fact, in other training runs of f_ϕ and MT model, we found other tokens that are spuriously correlated with the reward (including "conduct" – the token "conduct" has a high reward, but it introduces disfluency in the sentence).[3]

[3]Example generation 1: `the 66 countries and regions have been able to conduct the evidence in the dissemination of the virus in 2015 .` Example generation 2: `the some parents have been able to conduct the campaign day and the some comments on this matter and the many persons have been able to conduct attention .` The MT model integrates "conduct"

Experimental details on the above examples. For the Sudoku experiment, the hyperparameters are selected from the same sets as in §4.1. For the MT experiment, to train the classifier f_ϕ, the model is initialized by a WMT19 Zh-En MLE-trained model. Then, the source sentence is fed into the encoder, and the target sentence is fed into the decoder. However, we remove the attention mask in the decoder that prevents hidden states at token t from seeing future hidden states. The reward r_t is the probability that the t-th token is erroneous, according to f_ϕ. For D_{task}, our translation task uses the WMT19 Zh-En dataset, and f_ϕ is fine-tuned from an MLE-trained MT checkpoint using the WMT19 Zh-En dataset. We use a transformer model with 6 encoder layers and 6 decoder layers. The number of attention heads is 8 in both the encoder and the decoder. The FFN embedding dimension is 2048 in both the encoder and the decoder.

4.3 Covariate Shift

During RL training, the policy (i.e., the generator) may sample examples out of the support of the reward model. Therefore, in these examples, the reward model's behavior is underspecified – it may (or may not) assign high rewards to these low-quality examples.

Synthetic example: another Sudoku variant. $\mathcal{D}_{\text{reward}}$ contains 200k positive and 200k negative examples.[4] We design $\mathcal{D}_{\text{reward}}$ in such a way that the model behavior would be undefined for certain inputs. All examples end with 1; continuations that end with 2–9, are not in the support on the data used to train the reward function f_ϕ.

f_ϕ achieves 96.5% accuracy on the test split of $\mathcal{D}_{\text{reward}}$. We sample 1000 in-support (i.e., ending with 1) and 1000 out-of-support (i.e., ending with 2–9) invalid Sudokus. The model only misclassifies 1 out of 1000 example as valid on the in-support set; in contrast, 659 out of 1000 examples are misclassified as valid on the out-of-support set.

During RL training of the conditional text generation model, the reward for sampled generations increases above 0.8 – we expect the reward to imply that more than 80% continuations are estimated to be valid by the reward; however, only <10% of the continuations are actually valid.

in the generations but the use of "conduct" is incorrect and nonsensical.

[4]Negative examples are obtained by swapping two different tokens of a positive example 1–20 times.

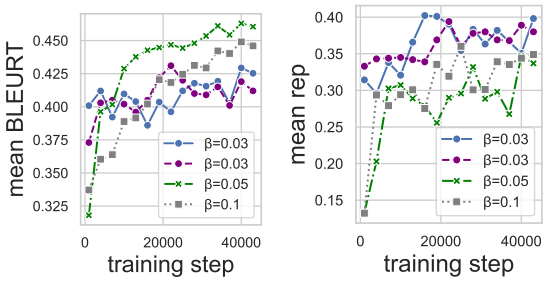

Figure 3: Left: mean BLEURT vs. training step. Right: mean rep vs. training step. Each point corresponds to the mean value for a bucket of 3,000 training steps. Each bucket contains ≥140 translations whose source sentences are longer than 180 tokens. We see that BLEURT increases during RL training; rep increases as well. rep for reference translations (whose source length >180) is 0.12, much smaller than achieved in our experiments. 93% of translations has rep <0.2. The two runs with $\beta = 0.03$ use different baselines (see §A). Repetition is a problem even for large β.

Real-world example 1: AgreeSum. One simple example reproduces the multi-doc AgreeSum summarization (Pang et al., 2021b). The input of the task is a cluster of articles, and the expected output is a summary that is faithful to *every* article in the cluster. We consider $\mathcal{D}_{\text{reward}}$ that consists of faithfulness annotations on article-summary pairs provided by the AgreeSum paper. The reward function f_ϕ is a summary-article faithfulness classifier. f_ϕ achieves 79% dev accuracy, which we use as the reward. However, the shortest summary in $\mathcal{D}_{\text{reward}}$ is 7-token-long, so the behavior of the reward for shorter summaries is underspecified. Training a summarizer using the faithfulness classifier as the reward leads to short summaries – most of which (>90%) are ≤ 2 tokens. Even though these near-empty summaries can be technically considered as being entailed in the article, we have not specified in $\mathcal{D}_{\text{reward}}$ that these summaries are acceptable.

Real-world example 2: MT using BLEURT. BLEURT (Sellam et al., 2020) is a metric trained on expert annotations provided by WMT metric tasks. We train a text generator by RL using BLEURT-20-D3, a distilled version of BLEURT-20. BLEURT is trained on very few repetitive generations. WMT15–19 human rating data (Stanojević et al., 2015; Bojar et al., 2016, 2017; Ma et al., 2018, 2019) are used to train BLEURT. We use BLEURT to train a MT model on the IWSLT14 De-En task (Cettolo et al., 2014). MLE-trained model achieves 63.9 in BLEURT on test set and

RL-trained model achieves 65.5, so RL is successful judging by the increase in BLEURT.

Repetitive translations are out-of-support in our case, where repetition is measured by rep defined to be the percentage of 3-grams that have appeared in the same translation before.[5] In fact, only 0.02% (58/247,157) translations have rep >0.4 and 0.05% translations have rep > 0.3 in $\mathcal{D}_{\text{reward}}$. In this example, we only analyze examples where the source sentence is >180 BPE-tokens.

We find that BLEURT does not punish for excessive repetition in the samples during RL: average BLEURT for translations with rep >0.4 (>40% of 3-grams are repetitions – an example is shown in the footnote to demonstrate that 40% is an undesirably large proportion)[6] in the first 45,000 steps of training[7] is 42.7, and average BLEURT for translations with rep <0.2 is 42.3.[8] So the reward does not discourage the MT model from generating repetitions.

Next, we show in Figure 3 that as training goes on, translations get more and more repetitive as BLEURT increases. To summarize, given that repetitive translations are rare in $\mathcal{D}_{\text{text}}$, the reward is underspecified on them. This repetition pattern is not discouraged by the reward, and thus it is subsequently exploited by the MT model.

Experimental details for the above examples. For AgreeSum, given URLs in the original dataset, to find the corresponding articles, we use the `newspaper3k` library. The reward function (classifier) is based on RoBERTa-large. The summarizer is based on BART-large (Lewis et al., 2020). For the MT experiment, the MT model has an embedding dimension of 512 for both the encoder and the decoder. The FFN embedding dimension is 1024 for both the encoder and the decoder. Both the encoder and the decoder have 4 attention heads and 6 layers. More details can be found in the appendix.

[5] We present an example of computing the metrics rep. The sentence 'a b c e d c e d c d' has rep $= 2/5 = 40\%$, given that among 'e d c,' 'd c e,' 'c e d,' 'e d c,' 'd c d,' two 3-grams are the same with the existing ones.

[6] As an example, the following sentence has rep $= 0.397$: `pip was adopted from "great expectations; superman was a foster child; and the azbeth salander," the girl with the dragon tattoo, "was a foster child and a pure man; lyra belacqua from philip pullman," and a foster child, jane eyre, adopted, and roald's james, and the great, and he was a parent, and a parent, and then, "and then, you know," and then, "and then, you know," and then, "and, you know," the "- and, you know," the "- and, you know," the "the" - and "you know," the "`

[7] Using $\beta = 0.05$ which leads to the best dev BLEURT.

[8] 0.2 is an acceptable threshold, given that 93% of translations whose source sentence length >180 have rep <0.2.

5 Possible Remedies

As discussed in §2, Skalse et al. (2022) has suggested that a pair of unhackable nontrivial original-proxy reward functions do not exist in theory. Then, when is it safe to use the proxy reward function? While this is still an open question, it is possible to reduce the extent of generating undesirable sentences through the following approaches.

The fundamental problem is that errors in the reward functions, specifically the over-confident errors where low-quality outputs have high rewards, can be exploited during RL training of text generators. Thus, one solution is to avoid OOD states that incur such errors by restricting the policy.

Restricting the policy by regularizing toward the ML solution. A common strategy is to regularize toward the ML solution. In practice, we can interpolate RL and ML losses (Wu et al., 2016), interleave RL and ML updates (Lewis et al., 2017; Guo et al., 2018), or use KL-regularized RL (Jaques et al., 2019; Stiennon et al., 2020; Ramamurthy et al., 2022).[9]

Here are a few potential issues. First, RL exploration could be important in case the reference dataset is small, and consequently, the ML solution is sub-optimal. For example, in AgreeSum, there are not enough reference summaries due to data collection costs, but given a decent article-summary faithfulness classifier, we can discover new summaries that have high rewards. Similarly, in creative generation tasks like story generation and textual style transfer, or in code generation, there may not be a large enough high-quality reference dataset, but a reward function is often available. Second, ML solution may not be optimal even with an adequately large reference dataset; e.g., degeneracies like unreasonably short translations (Stahlberg and Byrne, 2019; Kulikov et al.,

[9] Korbak et al. (2022) argue that KL-regularization is needed, because naïve RL would result in distributional collapse. They argue that RL is not the right framework for fine-tuning language models because of this reason, and we should look at Bayesian inference instead. But we argue that diversity (i.e., the ability to produce many different good generations) is not necessary in many conditional text generation tasks (as opposed to language modeling – unconditional generation). A perfect reward, even if resulting in less diverse generations, would still lead to good generations for tasks like MT. Thus, distributional collapse is not the problem, but reward gaming (which sometimes leads to collapse on undesirable cases) is, and the KL term is a potential remedy.

2021) and repetitive generations (Welleck et al., 2020b,a; Chiang and Chen, 2021) may often have high probabilities. Third, By relying on ML, we are optimizing toward a different objective; thus, we may need to find another automatic evaluator (instead of the proxy reward) to do hyperparameter tuning and model selection. Additionally, Gao et al. (2022) has discovered that larger coefficients for the KL penalty (for ML regularization) does not improve the frontier of the curve of the gold reward model score vs. KL divergence between the an RL-optimized model and the ML model.[10]

Restricting the policy by leveraging a discriminator. Following Goodfellow et al. (2014) and Pang et al. (2021b), another idea similar to ML-regularization is to leverage a discriminator that distinguishes between sampled generations and the set of dataset-provided generations.[11] During RL training, we force the model to produce generations that are indistinguishable from references according to the discriminator. Discriminator and RL updates are interleaved. It is difficult to use GAN to train a high-quality text generator, but we hypothesize that the discriminator can reduce easy-to-identify examples during RL training.

Fixing the reward itself. Another thread of remedies is to fix the reward itself. An effective approach is to **iteratively collect human annotations** (Stiennon et al., 2020; Bai et al., 2022; Fan et al., 2022): the reward is iteratively updated with human annotations on latest model generations; thus, the generations with low human preferences but high rewards will be corrected through this iterative process. One concern is the cost, which may prohibit an adequate amount of iterations or adequately frequent iterations. Krakovna et al. (2020) has discussed the possibility that a machine can learn to fool human evaluators in robotics, but it is unclear what the equivalence in conditional text generation is. So far, this approach has been successful, with the critical assumption that there is little budget/resource constraint to obtain enough annotations and enough iterations of annotations.

More discussion. An additional method in the RL literature is conservative Q learning (Kumar et al., 2020); it aims to push down all high rewards to ensure that the out-of-distribution states do not achieve high Q values, but the approach requires extensive hyperparameter tuning (Zheng et al., 2022). Another possibility to avoid the reward gaming issue is to avoid interaction with the environment, using methods like Pang and He (2021) to learn from demonstrations, so the errors in the reward function will be less exploited.

[10] See Figure 9 of Gao et al. (2022).

[11] The discriminator predicts whether the generation is machine-generated or comes from *the set of* references. This technique is useful when there are only few parallel datapoints.

6 Conclusion

To conclude, we use synthetic and real-world tasks to demonstrate that even if a learned reward achieves high performance on $\mathcal{D}_{\text{reward}}$, a high reward may still get assigned to undesirable patterns which get amplified during RL training of the conditional text generation model. A critical future direction is to investigate when or how easily a spurious feature could be exploited, by exploring the relationship among the minimum description length of a spurious feature (Voita and Titov, 2020) or similar statistics, the proportion of datapoints that contains the spurious feature, the choice of RL algorithm, and the degree of the reward gaming behavior. Additionally, further research on anti-gaming approaches is needed to fulfill the potential of training text generators by learned rewards.

Limitations

First, off-policy algorithms like Q learning are not explored in this discussion. Second, the reward gaming issue is not a novel topic in the RL community for tasks like gameplay or autonomous driving (Amodei et al., 2016; Koch et al., 2022). However, we hope to highlight issues in the NLG community especially given the recent endeavors on learning from learned metrics. In addition, the paper aims to demonstrate the existence of reward gaming in conditional text generation, not the certainty regardless of experimental settings (hyperparameters, architectures, etc.). Given that our experiments use reasonable settings which lead to degenerate texts, we argue that reward gaming could be a common issue when learning a text generation model using RL based on learned rewards, and the issue deserves attention from researchers and practitioners. We leave it to future work to investigate the *easiness* of reward gaming in practice, which is missing in this work.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, Online. Association for Computational Linguistics.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Chantal Amrhein and Rico Sennrich. 2022. Identifying weaknesses in machine translation metrics through minimum bayes risk decoding: A case study for comet. *arXiv preprint arXiv:2202.05148*.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710, Dublin, Ireland. Association for Computational Linguistics.

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

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2020. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Ondřej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the WMT17 metrics shared task. In *Proceedings of the Second Conference on Machine Translation*, pages 489–513, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. 2016. Results of the WMT16 metrics shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 199–231, Berlin, Germany. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, volume 57, Hanoi, Vietnam.

Ting-Rui Chiang and Yun-Nung Chen. 2021. Relating neural text degeneration to exposure bias. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 228–239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tom Everitt, Marcus Hutter, Ramana Kumar, and Victoria Krakovna. 2021. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *Synthese*, 198(27):6435–6467.

Xiang Fan, Yiwei Lyu, Paul Pu Liang, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2022. Nano: Nested human-in-the-loop reward learning for few-shot language model control. *arXiv preprint arXiv:2211.05750*.

Markus Freitag, George Foster, David Grangier, and Colin Cherry. 2020. Human-paraphrased references improve neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1183–1192, Online. Association for Computational Linguistics.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021a. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, George Foster, Alon Lavie, and Ondřej Bojar. 2021b. Results of the WMT21 metrics shared task: Evaluating metrics with expert-based human evaluations on TED and news domain. In *Proceedings of the Sixth Conference on Machine Translation*, pages 733–774, Online. Association for Computational Linguistics.

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling laws for reward model overoptimization. *arXiv preprint arXiv:2210.10760*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27.

Charles Goodhart. 1975. Problems of monetary management: the UK experience in papers in monetary economics. *Monetary Economics*, 1.

Chulaka Gunasekara, Guy Feigenblat, Benjamin Sznajder, Ranit Aharonov, and Sachindra Joshi. 2021. Using question answering rewards to improve abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 518–526, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Ferenc Huszár. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. 2018. Reward learning from human preferences and demonstrations in atari. *Advances in Neural Information Processing Systems*, 31.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*.

Zachary Kenton, Tom Everitt, Laura Weidinger, Iason Gabriel, Vladimir Mikulik, and Geoffrey Irving. 2021. Alignment of language agents. *arXiv preprint arXiv:2103.14659*.

Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael F Ong, Zhewei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmaszczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, et al. 2018. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 121–153. Springer.

Samuel Kiegeland and Julia Kreutzer. 2021. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1673–1681, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jack Koch, Lauro Langosco, Jacob Pfau, James Le, and Lee Sharkey. 2022. Objective robustness in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR.

Ryosuke Kohita, Akifumi Wachi, Yang Zhao, and Ryuki Tachibana. 2020. Q-learning with language model for edit-based unsupervised summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 470–484, Online. Association for Computational Linguistics.

Tomasz Korbak, Ethan Perez, and Christopher L Buckley. 2022. RL with KL penalties is better viewed as bayesian inference. *arXiv preprint arXiv:2205.11275*.

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. 2020. Specification gaming: the flip side of ai ingenuity. *DeepMind Blog*.

Ilia Kulikov, Maksim Eremeev, and Kyunghyun Cho. 2021. Characterizing and addressing the issue of oversmoothing in neural autoregressive sequence modeling. *arXiv preprint arXiv:2112.08914*.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. 2020. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34(4), pages 4501–4510.

Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Beslon, David M Bryson, et al. 2020. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artificial life*, 26(2):274–306.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453, Copenhagen, Denmark. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Arle Lommel, Hans Uszkoreit, and Aljoscha Burchardt. 2014. Multidimensional quality metrics (MQM): A framework for declaring and describing translation quality metrics. *Revista Tradumàtica: tecnologies de la traducció*, (12):455–463.

Qingsong Ma, Ondřej Bojar, and Yvette Graham. 2018. Results of the WMT18 metrics shared task: Both characters and embeddings achieve good performance. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 671–688, Belgium, Brussels. Association for Computational Linguistics.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 62–90, Florence, Italy. Association for Computational Linguistics.

Nitika Mathur, Johnny Wei, Markus Freitag, Qingsong Ma, and Ondřej Bojar. 2020. Results of the WMT20 metrics shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 688–725, Online. Association for Computational Linguistics.

Yuki Nakatani, Tomoyuki Kajiwara, and Takashi Ninomiya. 2022. Comparing BERT-based reward functions for deep reinforcement learning in machine translation. In *Proceedings of the 9th Workshop on Asian Translation*, pages 37–43, Gyeongju, Republic of Korea. International Conference on Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. 2022. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*.

Richard Yuanzhe Pang and He He. 2021. Text generation by learning from demonstrations. In *International Conference on Learning Representations*.

Richard Yuanzhe Pang, He He, and Kyunghyun Cho. 2021a. Amortized noisy channel neural machine translation. *arXiv preprint arXiv:2112.08670*.

Richard Yuanzhe Pang, Adam Lelkes, Vinh Tran, and Cong Yu. 2021b. AgreeSum: Agreement-oriented multi-document summarization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3377–3391, Online. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron Wallace. 2022. Combining feature and instance attribution to detect artifacts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1934–1946, Dublin, Ireland. Association for Computational Linguistics.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. *arXiv preprint arXiv:1608.07836*.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2022. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence, Italy. Association for Computational Linguistics.

Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, Hong Kong. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Thibault Sellam, Amy Pu, Hyung Won Chung, Sebastian Gehrmann, Qijun Tan, Markus Freitag, Dipanjan Das, and Ankur Parikh. 2020. Learning to evaluate translation beyond English: BLEURT submissions to the WMT metrics 2020 shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 921–927, Online. Association for Computational Linguistics.

Joar Skalse, Nikolaus HR Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking. *arXiv preprint arXiv:2209.13085*.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China. Association for Computational Linguistics.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the WMT15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273, Lisbon, Portugal. Association for Computational Linguistics.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12.

Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. 2019. Is deep reinforcement learning really superhuman on atari? leveling the playing field. *arXiv preprint arXiv:1908.04683*.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. 2020a. Consistency of a recurrent language model with respect to incomplete decoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5553–5568, Online. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020b. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Sarah Wiegreffe and Ana Marasovic. 2021. Teach me to explain: A review of datasets for explainable natural language processing. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

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

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

Qinqing Zheng, Amy Zhang, and Aditya Grover. 2022. Online decision transformer. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27042–27059. PMLR.

A More Background

For our policy gradient algorithms, we use the standard REINFORCE algorithm with tricks that are introduced in the following paragraphs.

Specifically, in all RL experiments, we first initialize the model using an MLE-trained model to ensure a good starting point for RL optimization. During training, we collect a set of trajectories through sampling from the current policy (i.e., generator). Then, we compute the estimated return $\hat{Q}_t$ at each time-step t.

Next, the estimated return $\hat{Q}_t$ is subtracted by a baseline. Therefore, the actual gradient update is as follows: $\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)[\hat{Q}(s_t, a_t) - b(s_t)]$, where $\hat{Q}(s_t, a_t) = \sum_{t'=t}^{T} r_{t'}$ assuming discount factor $\gamma = 1$, and b is possibly state-dependent. In particular, for Sudoku experiments as well as the experiment where we train an MT model using BLEURT as the reward, we attempt two variants of baseline: (1) using the average reward for the past 50 updates, which is an effective strategy in training models using sequence-level rewards (Kiegeland and Kreutzer, 2021), and (2) using the value function fitted by mean-squared error (so the estimated return subtracted by the value ends up being the advantage), introduced in full detail here.[12] For case (1), the results are shown in the blue lines in the plots; for case (2), the results are shown in the purple dotted lines in the plots. We use the Adam optimizer (Kingma and Ba, 2014) for all our experiments.

In particular, we use KL-regularized RL, as discussed in §3. Regularization toward ML may stabilize RL optimization, but it may still lead to higher rewards that correspond to undesirable behaviors, as discussed in §5. The coefficient for the KL term is tuned in {0.01, 0.05, 0.1} for Sudoku experiments and {0.01, 0.03, 0.05, 0.1, 0.25} for other experiments. For the purpose of this discussion, to illustrate the effect of reward gaming, the coefficient is tuned to achieve the highest validation reward; due to optimization issues in practice, a lower coefficient does not necessarily correspond to a higher reward. Larger coefficients may lead to lower proxy rewards but higher true rewards. While it may address the reward gaming problem in some experiments, we have shown in §4.3 that even large coefficients may lead to reward gaming.

Proximal policy optimization (PPO; Schulman et al., 2017) is a widely used algorithm that aims to avoid reward collapse. Our conclusion, however, does not depend on the RL algorithm. Using PPO prevents the optimization from converging to a very low reward, but it does not eliminate the possibility that high reward generations have undesirable patterns. In addition, Q learning, an off-policy RL algorithm that can leverage existing trajectories, is recently applied to also be applied in text generation (Kohita et al., 2020; Pang et al., 2021a) but is not explored in this discussion.

[12] https://spinningup.openai.com/en/latest/algorithms/vpg.html#pseudocode

B More Experimental Details

B.1 Details for the Experiments on Noise-Induced Spurious Correlation

Examples that are used to train the reward function. As explained in §4.1, there are 1M examples in total, 500k of which are positive examples and 500k are negative examples. The negative examples consist of the following parts: (i) 100k invalid Sudokus that are randomly sampled. None of the above examples end with 7. (ii) 100k invalid Sudokus obtained by removing l cells randomly from a random positive Sudoku, where l is an integer randomly sampled from 1 to 80. (iii) 300k invalid Sudokus that are obtained by swapping cell i and cell j of a random positive Sudoku; after swapping, we verify that the Sudoku is in fact invalid. The train/dev/test split of $\mathcal{D}_{\text{reward}}$ is 900k/50k/50k.

Reward. The RoBERTa-tiny-based (Liu et al., 2019) reward function has 4 encoder layers and 2 attention heads; the encoder embedding dimension is 64, and the dimension for FFN for 256. All the Sudoku-related experiments are done on either a single NVIDIA V100 GPU with 32G of memory or a single NVIDIA RTX 8000 GPU with 48G of memory. The reward training typically takes 1 hour. The batch size is tuned in {128, 256, 512}. The dropout rate is tuned in {0.01, 0.1}, and we find that 0.01 always works better. The max number of epochs is set to 60. The learning rate is tuned in {1e-4, 5e-4, 1e-3}. For the best configuration, we use batch size 512 and learning rate 5e-4. It achieves a 99.3% accuracy on the dev set (5% split), and a 99.3% accuracy on the test set (5% split).

Out of 1000 samples of invalid Sudokus that end with 7 and contain 81 tokens, the trained classifier predicts (incorrectly) that 1000 are valid. Out of 1000 samples of invalid Sudokus that end with 7 and contain fewer than 81 tokens, the trained clas-

sifier predicts that 0 is valid. The performance of Sudokus longer than 81 tokens is irrelevant, given that during RL sampling as well as during generation test time, the sequences are constrained such that they can at most generate $81 - k$ tokens where k is the length in the given source sequence.

Sequence generator. Suppose the input to the generator contains k numbers. During RL sampling and during test-time of the generator, the sequence generator is constrained to generate at most $81 - k$ numbers. However, it can generate fewer than $81 - k$ numbers. To avoid sequence generators from generating overly short continuations, part (ii) of the negative examples, described above, contains examples that are too short.

For the sequence generator, we use a smaller version of the `transformer_iwslt_de_en` architecture in `fairseq` (Ott et al., 2019). The encoder embedding dimension and the decoder embedding dimension are both 32. We use 2 attention heads in both the encoder and the decoder. The dimension for FFN in both the encoder and the decoder is 64. There are 2 encoder layers and 2 decoder layers. All the text generation models in the Sudoku experiments have 43k parameters.

The batch length (i.e., number of tokens in a batch) is tuned in {8192, 16,384, 32,768, 65,536}. The learning rate is tuned in {1e-4, 1.5e-4, 2e-4}. The dropout rate is tuned in {0.01, 0.1, 0.3}. For optimal reward, we choose a batch length of 32,768, a learning rate of 1.5e-4, and a dropout rate of 0.01. The training algorithm is detailed in §A.

B.2 Details for the Experiments on Naturally Occurring Spurious Correlations

Sudoku revisited. For the second Sudoku example, the hyperparameters are selected from the same sets as in §B.1. For the best-performing classifier, the learning rate is 5e-4 and the dropout rate is 0.01. For the sequence generator, we use the same hyperparameters as before. The lack of repetition in the last nine numbers (of the output) is spuriously correlated with a high reward, given that non-repetition is a necessary but not sufficient condition for a valid Sudoku. f_ϕ achieves 99.9% accuracy on test set of $\mathcal{D}_{\text{reward}}$. The text generator learns to exploit the non-repetition pattern which leads to high rewards, but the generations are mostly wrong.

Training an MT model using the WMT MQM dataset. To train the reward function, the learning rate is selected from {1e-4, 2e-4, 5e-4}, and dropout is selected from {0.01, 0.1, 0.3}. For optimal performance, we use a learning rate of 2e-4 and a dropout rate of 0.3. Training the reward function takes around 3 hours.

For D_{task}, our translation task uses the WMT19 Zh-En dataset, and f_ϕ is fine-tuned from an MLE-trained MT checkpoint using the WMT19 Zh-En dataset. We use a transformer model with 6 encoder layers and 6 decoder layers. The number of attention heads is 8 in both the encoder and the decoder. The FFN embedding dimension is 2048 in both the encoder and the decoder. There are 82.6M parameters in the model.

The algorithm is detailed in §A. We use a KL coefficient of 0.1. We use a dropout rate of 0.3, a learning rate of 1e-4, and a batch length of 4096. All the MT experiments are done on a single NVIDIA RTX 8000 GPU with 48G of memory. Training time is only 24 hours, given that we do not need to train the model till convergence to see the undesirable patterns in generations.

B.3 Details for the Experiments on Covariate Shift

Experimental details for AgreeSum. Given URLs in the original dataset, to find the corresponding articles, we use the `newspaper3k` library. We use slightly different architectures from the AgreeSum paper. The reward function (classifier) is based on RoBERTa-large (Liu et al., 2019) with 355M parameters. We use a learning rate of 5e-4, a dropout rate of 0.1. The submitted job for the classifier is 24-hour-long. The summarizer is based on BART-large (Lewis et al., 2020) with 406M parameters. We use a learning rate of 3e-5, a batch length of 2048, and a dropout rate of 0.1. We use a single NVIDIA RTX 8000 GPU for AgreeSum experiments.

Experiment Details for MT with BLEURT as Reward. The BLEURT-20-D3 evaluator has around 30M parameters. For the MT model that is trained on the IWSLT14 De-En dataset (train/dev/test size: 160,239/7,283/6,750), the embedding dimension is 512 for both the encoder and the decoder. The FFN embedding dimension is 1024 for both the encoder and the decoder. Both the encoder and the decoder have 4 attention heads and 6 layers. There are 39.5M parameters in the model. The learning rate is selected from {1e-4, 3e-4}. The batch length (i.e., number of tokens in a batch) is set to be 4,096 and the dropout rate is set to be 0.3 – these are the

optimal choices for IWSLT14 De-En experiments trained using MLE. KL coefficient is selected from in $\{0.01, 0.03, 0.05, 0.1\}$. We choose the hyperparameter settings that lead to the highest validation BLEURT. Training time is around 20 hours on a single NVIDIA RTX 8000 GPU.