General Paper

Neural Text Generation with Artificial Negative Examples to Address Repeating and Dropping Errors

Keisuke Shirai[†], Kazuma Hashimoto*[††], Akiko Eriguchi*[†††], Takashi Ninomiya[††††] and Shinsuke Mori[†††††]

Neural text generation models that are conditioned on a given input (e.g., machine translation and image captioning) are typically trained through maximum likelihood estimation of the target text. However, models trained in this manner often suffer from various types of errors when making subsequent inferences. In this study, we propose suppressing an arbitrary type of error by training the text generation model in a reinforcement learning framework; herein, we use a trainable reward function that can discriminate between references and sentences, containing the targeted type of errors. We create such negative examples by artificially injecting the targeted errors into the references. In the experiments, we focus on two error types; repeated and dropped tokens in model-generated text. The experimental results demonstrate that our method can suppress generation errors, and achieves significant improvements on two machine translation and two image captioning tasks.

Key Words: *Machine Translation, Image Captioning, Discriminator, Negative Example*

1 Introduction

Conditional neural text generation models are expected to generate human-readable text that accurately describes information from a given source (Sutskever et al. 2014; Vinyals et al. 2015). These models are trained in a supervised fashion, by providing them with ground-truth symbols (Williams and Zipser 1989). Conversely, when making inferences (i.e., during testing), the models usually generate text in a left-to-right fashion; notably, these text generation models suffer from various generation errors. For example, the models unnecessarily repeat tokens, and will drop or lose informative tokens. We refer to these two error types as *repeating* and *dropping errors.* In this study, our goal is to design a training framework that could explicitly suppress a specific

[†] Graduate School of Informatics, Kyoto University
[††] Salesforce Research
[†††] Microsoft Research
[††††] Graduate School of Science and Engineering, Ehime University
[†††††] Academic Center for Computing and Media Studies, Kyoto University
*Work performed while the authors were at the University of Tokyo.

type of generation error.

We focus on a reinforcement learning (RL) framework proposed by Ranzato et al. (2016), and incorporate a reward function to penalize erroneous text generation during training. The RL framework has recently been applied to several text generation tasks, and research has been conditioned on the design of task-oriented reward functions (Wu et al. 2016; Zhang and Lapata 2017; Rennie et al. 2017). More recent studies (Dai et al. 2017; Gu et al. 2018) have proposed training a discriminative reward function (i.e., a discriminator) in generative adversarial network frameworks. However, these reward functions are not designed to handle a specific type of generation error.

To answer our research question, we propose a new RL framework that suppresses an arbitrary type of generation error. Figure 1 illustrates the overview of our method. First, we train a reward function that discriminates between references and sentences, containing the targeted type of error. To train such a discriminator, we introduce artificially generated negative examples by injecting the specific type of error into the references (Figure 1 (a)). The trained discriminator is then expected to capture the targeted errors according to Ranzato et al. (2016), a text generation model learns to generate text while suppressing the specified error type (Figure 1 (b)). In this study, we consider two error types: repeating and dropping errors. We provide examples of these error types in a translation task in Table 1. We demonstrate that our method can suppress these errors and improve the generation performance on two tasks each for translation and captioning.

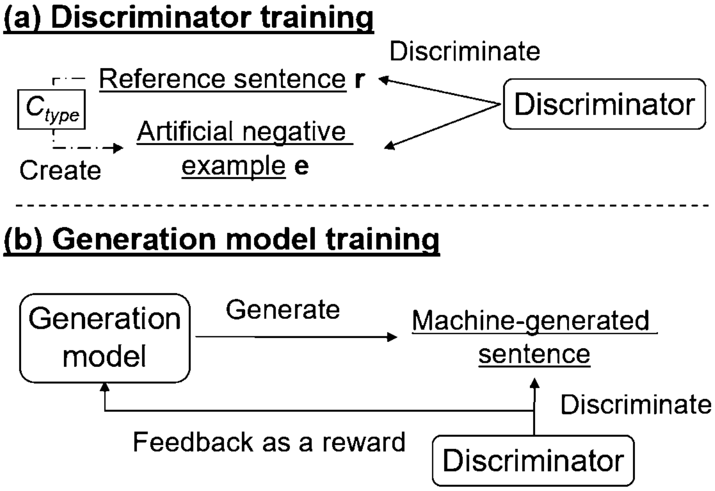


Fig. 1 The overview of our method.

Repeating error example	
Source	幽門形成術を行わない分節胃切除術は術後のＱＯＬの点で優れていた
Reference	Segmental gastric resection without pyloroplasty was superior in terms of QOL.
Model output	It was concluded that segmental gastrectomy without pyloroplasty was superior to postoperative quality of life(QOL) in terms of postoperative quality of life(QOL).
Dropping error example	
Source	ＨＭ１８は加速粒子がＨ，Ｄに限られ，主に１１Ｃ，１８ＦなどＰＥＴ用核種の製造に利用される
Reference	Particles accelerated in HM18 are limited to H and D, and which are mainly used for preparation of nuclides for PET such as 11C and 18F.
Model output	HM18 is mainly used for the production of positron nuclides such as 11C, 18F, etc.

Table 1 Error generation examples of neural text generation model in a Japanese-to-English translation task. The tokens with a solid line are repeated by the generation model. The tokens with a dashed line are dropped from a source sentence at the decoding step.

Our contributions are three-fold.

- We propose a novel RL framework that suppresses text generation errors by specifically targeting an error type. We show that our method can suppress two types of generation errors: repeating and dropping errors.
- Our method can be expansively applied to the existing RL framework using other functions, such as GLEU, to further boost its generation performance.
- Our analyses show that the proposed method functions more effectively then traditional methods do, as the training example size becomes smaller and more generation errors appear.

2 Neural Text Generation

2.1 Maximum likelihood training

In conditional neural text generation models, an encoder encodes the source-side information s (e.g., a source sentence in machine translation (Sutskever et al. 2014) or an image in image captioning (Vinyals et al. 2015)) into an intermediate representation h^s, and then, a decoder is trained to generate a reference $r = (r_1, r_2, \cdots, r_n)$ conditioning on h^s. To train the model in a supervised manner, we follow the maximum likelihood estimation (MLE) objective, as

$$L_{\text{MLE}} = -\sum_{j=1}^{n} \log p(r_j | r_{<j}, s). \tag{1}$$

In this study, we focus on two generation tasks: machine translation and image captioning.

At the time of inference, the text generation model generates a target-side sentence, $t = (t_1, t_2, \cdots, t_{n'})$. When generating the j-th token, the model follows a categorical distribution, $p(t_j | t_{<j}, s)$, to generate a token. To generate a better sentence, we use various decoding techniques, including beam search.

2.2 Reinforcement learning

Next, we describe the RL framework proposed in Ranzato et al. (2016). When training the text generation model in the RL framework, we consider the generation model as an agent and the categorical distribution at the j-th decoding step, $p(t_j | s, t_{<j})$, as a policy; here, t_j is the j-th token generated by the model. We then choose a token from the categorical distribution as an action. After generating a complete target sentence, $t = (t_1, t_2, \cdots, t_m)$, REINFORCE (Williams 1992) was used to define the loss function, as follows.

$$L_{\mathrm{RL}} = -\sum_{j=1}^{m} \{\log p(t_j | s, t_{<j})(R(s,t) - b(s, t_j))\}, \tag{2}$$

where R is a reward function, and b is a baseline network (Sutton and Barto 2018) that is used to reduce the variance of the gradients.[1] However, using only Equation (2) causes the training to become unstable (Wu et al. 2016). Thus, the following joint loss function was used.

$$\lambda_{\mathrm{MIXED}}\ L_{\mathrm{MLE}} + (1 - \lambda_{\mathrm{MIXED}})\ L_{\mathrm{RL}}, \tag{3}$$

where λ_{MIXED} is a hyperparameter that controls the strength of the two signals.

2.3 Generation errors

Neural text generation models suffer from various types of errors, such as repeating, dropping, grammatical, and mis-ordering errors. Among them, repeating and dropping errors have been most frequently addressed in previous research (Mi et al. 2016; Malaviya et al. 2018; Holtzman et al. 2020; Welleck et al. 2020). As a preliminary experiment, we trained a neural machine translation model with the MLE objective on the WAT'15 Workshop on Japanese-to-English translation tasks (Nakazawa et al. 2016); additionally, we manually checked 100 examples that were randomly sampled from the translation results on the test dataset. We found that 17 sentences contained repeating errors, wherein some tokens were repeated unnecessarily, and 13 sentences contained dropping errors, wherein some tokens were dropped from their reference

[1] Based on Ranzato et al. (2016), we used a linear regression model as the baseline network.

sentences.[2] We also observed that the model trained with the widely used RL approach in Wu et al. (2016) improve the BLEU scores (Papineni et al. 2002), but did not specifically suppress the above two types of errors.[3] Based on these observations, we aimed to improve the generation performance of the text generation model by suppressing these two common types of errors.

3 Approach

We propose to suppress the targeted type of errors using the RL framework. Our main objective was to design an R in Equation (2) that provides negative rewards to the generated text, if the text contains the targeted type of errors, and positive rewards to the text otherwise. We used a discriminator to instantiate such an R function. We trained the discriminator by taking references as positive examples and erroneous sentences as negative examples. We prepared these negative examples by artificially injecting the targeted errors into the references. We describe the negative examples in Section 3.1 and the discriminator in Section 3.2.

3.1 Artificial negative examples

Our aim was to prepare a discriminator that is capable of specifically focusing on one error type. The discriminator requires negative examples, such that they always contain the targeted type of error. We created such negative examples directly from the references using an error-generating function, $e = C_{type}(r)$; here, e is a sentence containing an error of a specific *type*, and r is a reference. We designed C_{type} depending on the error type on which we focused. We refer to e as an artificial negative example (ANE).

In this study, we designed two types of ANEs to handle repeating and dropping errors. Table 2 shows a reference sentence and its negative examples for each of the two types.[4] In the following section, we describe the design of the ANEs.

Artificial repeating sentence We created ANEs for repeating errors by modifying the reference sentences to repeat their tokens. In general, such repeated tokens do not always appear consecutively in model-generated sentences; therefore, we propose the C_{type} function as follows. Given a reference r with length n, $C_{repeat}(r)$ returns an artificial repeating sentence by duplicating i consecutive tokens, starting from the j-th token, at k randomly selected positions. We randomly

[2] These 100 translation examples are sampled from the RNN-based NMT (RNMT) result in Section 5.1. The error examples in Table 1 were sampled from these translation results.

[3] Specific details are provided in Section 5.2.

[4] Although the tokens were split at a word-level in this example, we applied our method at the subword-level in our experiments.

Reference	A man is known by the company he keeps .
Artificial repeating sentence	A man is known by the company by the company he keeps by the company .
Artificial dropping sentence	A man is known he keeps .

Table 2 Examples of artificial negative examples. Three consecutive tokens "by the company" are repeated in the repeating example, while they are dropped in the dropping example.

chose i from $(1, 2, \cdots, m_{\text{rep}})$, j from $(1, 2, \cdots, n-i+1)$, and k from $(1, 2, \cdots, n_{\text{rep}})$ for each example; here, m_{rep} is the maximum number of consecutive tokens, and n_{rep} is the maximum number of duplications. The k random positions were selected from $(1, 2, \cdots, j-1, j+i-1, \cdots, n-1, n)$ to avoid breaking the original consecutive tokens. In Table 2, three consecutive tokens "by the company," ($i = 3$) starting from the 5-th token ($j = 5$), were repeated twice ($k = 2$), and were inserted after the 7-th token, "company," and the 9-th token, "keeps," of the reference. We set the hyperparameters $(m_{\text{rep}}, n_{\text{rep}}) = (4, 4)$. We instead set $m_{\text{rep}} = n$ if n was smaller than n_{rep}.

Artificial dropping sentence We created ANEs for dropping errors by modifying the reference sentences to drop consecutive tokens. Given a reference r with length n, $C_{drop}(r)$ returns an artificial dropping sentence by dropping i consecutive tokens, starting from the j-th token. We randomly chose i from $(1, 2, \cdots, m_{\text{drop}})$ and j from $(1, 2, \cdots, n-i+1)$ for each example, where m_{drop} is the maximum number of consecutive tokens. In the example that is presented in Table 2, three consecutive tokens ($i = 3$), starting from the 5-th token ($j = 5$), were dropped. We set the hyperparameter $m_{\text{drop}} = 4$. This setup allowed us to drop i randomly chosen consecutive tokens ($i = 1, 2, \cdots, 4$). We instead set $m_{\text{drop}} = n$ if n was smaller than m_{drop} and larger than 2. C_{drop} returns an end-of-sentence token if n equals to 1.

3.2 Discriminator

We trained the discriminator using references and their ANEs. Our discriminator was a binary classifier that accepted a pair of source s and its target sentence (either r or $e = C_{type}(r)$) as the input and a real value in [0, 1] as the output. We minimized the following loss function as follows.

$$L_{\text{DIS}} = -\mathbb{E}_{s,r}[\log \mathrm{D}(s,r)] - \mathbb{E}_{s,e}[\log(1-\mathrm{D}(s,e))], \tag{4}$$

where D denotes the discriminator. During the discriminator training, we called $C_{type}(r)$ every time we processed r in a mini-batch. Once the discriminator training was completed, we used the trained discriminator D as $R(s,t)$ in Equation (2) by freezing its parameters, such that $R(s,t)$

could output a reward value for a generated sentence t to train the text generation models.

Our discriminator D comprises two types of encoders: source-side and target-side. The source-side encoder encodes source-side information s into a fixed-size vector h^s. The target-side encoder receives h^s, and encodes the target sentence $t = (t_1, t_2, \cdots, t_n)$ into a sequence of representations $H^t = (h^t_1, h^t_2, \cdots, h^t_n)$. Once H^t is calculated, we compute $\hat{h}^t$ as $\hat{h}^t = maxpool(H^t)$. The discriminator finally obtains output y, as

$$y = f_{\text{sigmoid}}(W_o f_{\text{ReLU}}(W_h \hat{h}^t + b_h) + b_o).$$

Here, $W_h \in \mathbb{R}^{d_h/2 \times d_h}$, $b_h \in \mathbb{R}^{d_h/2}$, $W_o \in \mathbb{R}^{1 \times d_h/2}$, and $b_o \in \mathbb{R}^1$ are learnable parameters, and d_h is the dimension of h^t. f_{ReLU} is the rectified linear unit (ReLU), and f_{sigmoid} is the logistic sigmoid function.

4 Experimental Settings

4.1 Datasets

Regarding machine translation, we conducted experiments on Japanese-to-English (Ja-En) and German-to-English (De-En) tasks. For the Ja-En task, we used the Asian scientific paper excerpt corpus (ASPEC) from WAT'15. Specifically, we used `train-1.txt` and `train-2.txt` for training, `dev.txt` for development, and `test.txt` for testing, according to Hashimoto and Tsuruoka (2019). Both Japanese and English sentences were preprocessed as recommended in WAT'15.[5] For the De-En task, we used the datasets provided by WMT'16[6] as well as all parallel corpora (`Europarl v7`, `Common Crawl corpus`, and `News Commentary v11`) for training; NewsTest2013 for development; and NewsTest2014 for testing.

Regarding image captioning, we conducted experiments on two datasets: MS COCO (Lin et al. 2014) and Flickr30K (Plummer et al. 2015). For training, developing, and testing, we followed the splits provided by Karpathy and Fei-Fei (2015).

We used `SentencePiece` (Kudo and Richardson 2018) to tokenize sentences and build vocabularies. We empirically chose a vocabulary size of $16,000$ for each language in the Ja-En translation task; $16,000$ for both languages in the De-En translation task, because these languages share an alphabet (Sennrich et al. 2016); and $2,000$ for the MS COCO and Flickr30K captioning tasks. The vocabulary also contained three special tokens: $\langle \text{s} \rangle$, which signaled the

[5] `http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2015/baseline/dataPreparationJE.html`

[6] `http://www.statmt.org/wmt16/translation-task.html`

Dataset	Vocabulary size		# examples		
	Source	Target	Train	Dev	Test
WAT'15	16,000	16,000	2,000,000	1,790	1,812
WMT'16	16,000		4,548,880	3,000	3,003
MS COCO	——	2,000	82,787	5,000	5,000
Flickr30K	——	2,000	29,000	1,014	1,000

Table 3 Dataset statistics and vocabulary sizes $|V|$.

beginning, ⟨/s⟩, which signaled the end of a sentence, and ⟨unk⟩ for out-of-vocabulary tokens. Table 3 shows the statistics regarding the different datasets. During training in machine translation, we removed all empty sentences and any sentence pairs, whose maximum length was longer than 80, from the training dataset.

4.2 Models

4.2.1 Machine translation

Text generation model For the translation tasks, we used two types of translation models: RNMT model (Bahdanau et al. 2015) and the Transformer (Vaswani et al. 2017). Our RNMT model is an attention-based NMT model with a 2-layer bidirectional long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) encoder and a 2-layer unidirectional LSTM decoder. We used the attention mechanism proposed by Bahdanau et al. (2015). The hidden state size and embedding size were both set to 512.

Our Transformer comprises a 6-layer encoder and 6-layer decoder. The hidden state size and number of heads were set to 512 and 8, respectively. To add positional information, we used a positional encoding technique with sine and cosine functions, as proposed by Vaswani et al. (2017).

Discriminator For the discriminator, we used a 2-layer unidirectional LSTM for the source-side and target-side encoders, with $d_h = 512$.

4.2.2 Image captioning

Text generation model For the image captioning tasks, we used a simple show-and-tell model (Vinyals et al. 2015). We used ResNet-152 (He et al. 2016), which was pre-trained on an ImageNet classification dataset (Russakovsky et al. 2015), as an encoder by freezing its model parameters to only extract features, and used a 512-dimensional 1-layer unidirectional LSTM as a decoder. Once an image feature $f^i \in \mathbb{R}^{2048}$ is extracted, the decoder state h_0^t is initialized as $h_0^t = W_i f^i + b_i$; here, $W_i \in \mathbb{R}^{512\times 2048}$ is a weight matrix, and $b_i \in \mathbb{R}^{512}$ is a bias vector.

Discriminator For the discriminator, we projected the feature vector f^i into a fixed-size vector h^s as $h^s = W_D f^i + b_D$, where the weight matrix $W_D \in \mathbb{R}^{512 \times 2048}$ and a bias vector $b_D \in \mathbb{R}^{512}$ are learnable parameters. We used a unidirectional LSTM as the target-side encoder, with $d_h = 512$.

4.3 Training strategies

4.3.1 Training discriminators

We used Adam (Kingma and Ba 2014) with an initial learning rate of 1.0×10^{-3} and a weight decay at a rate of 1.0×10^{-6}. We assessed its accuracy on the development dataset every $1,000$ iterations, and halved the learning rate when the accuracy worsened. The training was concluded after the learning rate was halved five times. We chose the best models based on those with the best accuracy with the development dataset. For the development dataset, we created one negative example for each reference sentence.

4.3.2 Training text generation models

The training of the text generation models can be divided into two steps: a pre-training step with MLE loss Equation (1) and reinforcement learning step with RL loss Equation (3) and the pre-trained model, following the steps outlined in Wu et al. (2016).

In the pre-training step, we used Adam with an initial learning rate of 1.0×10^{-3} and a weight decay with a rate of 1.0×10^{-6}, unless stated otherwise. Each mini-batch contained 128 examples. We assessed its perplexity on the development dataset every $1,000$ iterations, and halved the learning rate if the perplexity worsened. We finished the training when we halved the learning rate five times. For the Transformer, we used AdamW (Loshchilov and Hutter 2019) with a weight decay with a rate of 1.0×10^{-4}. When using AdamW, we scheduled the learning rate as follows.

$$lr = \begin{cases} lr_{\text{ini}} + step \times \frac{lr_{\text{max}} - lr_{\text{ini}}}{N_{\text{wm}}} & \text{if } step \leq N_{\text{wm}} \\ lr_{\text{max}} \times \eta(step) & \text{otherwise} \end{cases}, \tag{5}$$

where

$$\eta(step) = 0.5 + 0.5 \times \cos(\pi \times \frac{step - N_{\text{wm}}}{N_{\text{wm}} \times N_{\text{cl}}}).$$

This scheduling comprised two steps: a linear warmup step from lr_{ini} to lr_{max}, for the first N_{wm} iterations, and a cosine annealing step from lr_{max} to 0, for $N_{\text{wm}} \times N_{\text{cs}}$ iterations. We used empirically tuned hyperparameters as $(lr_{\text{ini}}, lr_{\text{max}}, N_{\text{wm}}, N_{\text{cs}}) = (5.0 \times 10^{-6},\ 5.0 \times 10^{-4},\ 4,000,\ 24)$. Each mini-batch contained 512 examples for the Transformer. For the pre-training step, we clipped the gradients (Pascanu et al. 2013) with a value of 1.0, and used the label-smoothing technique (Szegedy et al. 2016) with a rate of 0.1. We chose the best models based on the lowest perplexity with the development dataset.

During the reinforcement learning step, we used a stochastic gradient descent with momentum. For the translation tasks, we tuned the learning rate of each model with a momentum rate of 0.9. For the captioning tasks, we consistently used a learning rate of 5.0×10^{-2} with a momentum rate of 0.9. During this step, we extracted $20,000$ iterations to fine-tune the parameters of the generation model. We continued to use the same gradient clipping and label-smoothing techniques as outlined above. Each mini-batch contained 64 examples. In particular, we carefully tuned λ_{mixed} in Equation (3).[7] We report the inferences of our model as generated by a beam search with a width of 10.

4.4 Evaluation metrics

One goal of this study is to suppress the frequency of repeating and dropping errors. To quantitatively evaluate these errors in the model's output, we used REP and DROP scores (Malaviya et al. 2018). In this section, we first describe these two metrics in detail, and then, introduce our task-specific metrics.

4.4.1 REP score

The REP score calculates the number of n-gram repetitions that are included in a model-generated sentence t, given its reference r, as follows.

$$\mathrm{REP}(r,t) = \frac{\sigma(t,r)}{\sum_{w \in V} r(ww) + \sum_{s \in V_r^n} r(s)},$$

where

$$\begin{aligned}\sigma(t,r) =& \lambda_2 \sum_{s \in V_r^n, t(s) \geq 2} \max\{0, t(s) - r(s)\} \\ &+ \lambda_1 \sum_{w \in V} \max\{0, t(ww) - r(ww)\}. \end{aligned} \tag{6}$$

V_r^n is the set of all n-grams included in the reference. $r(ww)$ and $r(s)$ indicate the frequency of consecutive 1-grams w and n-grams s of the reference, respectively, and $t(ww)$ and $t(s)$ are those of the machine-generated sentence t. λ_1 and λ_2 are hypeparameters.

[7] Specifically, we searched the values in a coarse-to-fine fashion in $\{0.5, 0.3, 0.1, 7.5\times10^{-2}, 5.0\times10^{-2}, 2.5\times10^{-2}, 1.0 \times 10^{-2}, 7.5 \times 10^{-3}, 5.0 \times 10^{-3}, 2.5 \times 10^{-3}$, and $1.0 \times 10^{-3}\}$.

The REP score is defined for each n-gram separately; however, in this study, we propose using an extended REP (eREP) score that evaluates consecutive 1-gram and n-grams ($n = 2,\ 3,\ 4$). The eREP score is calculated as follows.

$$\mathrm{eREP}(r, t) = \frac{\sigma(t, r)}{\sum_{w \in V} t(ww) + \sum_{s \in V_t^n} t(s)},$$

where

$$\begin{aligned}\sigma(t, r) = &\sum_{n=2}^{4} \lambda_n \sum_{s \in V_t^n, t(s) \geq 2} \max\{0, t(s) - r(s)\} \\ &+ \lambda_1 \sum_{w \in V} \max\{0, t(ww) - r(ww)\}.\end{aligned}$$

V_t^n is the set of all n-grams included in the machine-generated sentence t. We weight the repetitions of n-grams equally ($\lambda_n = 1$). For the eREP score, a lower value is better.

4.4.2 DROP score

For the machine translation tasks, the DROP score calculates the number of source tokens that are *not* covered (i.e., included) in the model-generated sentences. A word alignment tool is used to identify which source tokens are aligned with their referent, and to calculate the ratio of those not aligned with the generated text (hypothesis). In other words, this study aims to evaluate how well the source information is covered, and to assess if the DROP score is defined as follows:

$$\mathrm{DROP}(c_{\mathrm{ref}}, c_{\mathrm{hyp}}) = 1 - \frac{1}{|c_{\mathrm{ref}}|} \sum_{i \in c_{\mathrm{ref}}} in(i),$$

where c_{ref} and c_{hyp} represent the set of source token indices in the source-reference and source-hypothesis alignments, respectively. $in(i)$ is a function that returns 1 if an index i is included in c_{hyp} and 0 otherwise. For the DROP score, a lower score is better, as is the case for the eREP score.

For the captioning tasks, however, the DROP score was not applicable, because no source sentence exists. For our experiment, we instead used ROUGE$_L$ (Chen et al. 2015) to evaluate the number of dropping errors that were found in the model's output. In ROUGE$_L$, contrary to the DROP score, a higher score is better. During our preliminary experiment, to assess the relationship between the DROP score and ROUGE$_L$, we calculated the Pearson correlation coefficient between ROUGE$_L$ and DROP on the RNMT's WAT'15 Ja-En result. We confirmed a negative moderate correlation (-0.430) between these two metrics, suggesting that we can use

ROUGE$_L$, instead of the DROP score, for the captioning tasks.

4.4.3 Task-specific metrics

We used BLEU and METEOR (Malaviya et al. 2018) for the translation tasks, and BLEU and CIDEr (Vedantam et al. 2015) for the captioning tasks. For the captioning tasks, we used publicly available code[8] to calculate the BLEU and CIDEr scores. Note that, for the captioning tasks, we report BLEU-1,2,3,4.

4.5 Model configurations

– **MLE** is the baseline model trained by the MLE loss in Equation (1).

– **RL-D$_{\text{REP}}$, RL-D$_{\text{DROP}}$** are our proposed models trained by the RL framework in Equation (3) with our proposed discriminator. The text generation model parameters were initialized using the MLEbaseline. RL-D$_{\text{REP}}$ and RL-D$_{\text{DROP}}$ were trained to suppress repeating and dropping errors with discriminators D$_{\text{REP}}$ and D$_{\text{DROP}}$, respectively.

5 Results

We report the scores in the range of [0, 100]. The symbol † follows a score if the system produced a significant improvement against MLE ($p < 0.05$). In our experiments, a statistical significance test was performed using the paired bootstrap resampling method (Koehn 2004).

5.1 Suppressing the targeted errors

In this section, we present the results obtained using the discriminator to suppress repeating and dropping errors. For the RL-D$_{\text{REP}}$ and RL-D$_{\text{DROP}}$ models, we chose the best models based on the scores from the development datasets that we would like to improve (e.g., the eREP score was used for RL-D$_{\text{REP}}$).

We first present the accuracy of our discriminators because the discriminator plays a key role in our method. Table 4 reports the binary classification accuracy for the development datasets. D$_{\text{REP}}$ achieved over 96% accuracy for all of the tasks, whereas the accuracy of D$_{\text{DROP}}$ was not as high as that of D$_{\text{REP}}$. A possible reason for this is that identifying the dropped tokens in a sentence is an inherently more difficult task than identifying the n-gram repetitions.

Table 5 shows the main results from the translation tasks. RL-D$_{\text{REP}}$ consistently improved

[8] `https://github.com/tylin/coco-caption`

its eREP scores, with the exception that the MLE-based Transformer in the De-En task had much less room for improvement. RL-$\mathrm{D}_{\mathrm{DROP}}$ also consistently improved its DROP scores for both the RNMT and Transformer models. When receiving rewards from the discriminator, namely $\mathrm{D}_{\mathrm{REP}}$ and $\mathrm{D}_{\mathrm{DROP}}$, the text generation models learned to suppress the repeating and dropping errors. Note also that RL-$\mathrm{D}_{\mathrm{REP}}$ tended to increase the BLEU and RL-$\mathrm{D}_{\mathrm{DROP}}$ tended to increase the METEOR, with some exceptions. For the former, we consider that reducing the number of repetitions using RL-$\mathrm{D}_{\mathrm{REP}}$ led to higher n-gram precision; thus, the model was able to achieve higher BLEU scores. For the latter, this is presumably because METEOR is based on unigram precision and recall, and thus, places more weight on recall than precision. Therefore, suppressing the dropping errors and restoring dropped tokens using RL-$\mathrm{D}_{\mathrm{DROP}}$ contributed to the improvements in the METEOR score.

Table 6 shows the results from the image captioning tasks. Again, note that RL-$\mathrm{D}_{\mathrm{REP}}$ significantly improved the eREP scores, leading to better BLEU-1,2,3,4 scores for both the COCO and Flickr30K tasks. Alternatively, RL-$\mathrm{D}_{\mathrm{DROP}}$ did not achieve significant improvements in ROUGE$_L$. One possible reason arises from the diversity of the text in the captioning task (Dai et al. 2017).

Task	$\mathrm{D}_{\mathrm{REP}}$	$\mathrm{D}_{\mathrm{DROP}}$
WAT'15 Ja-En	96.34	77.60
WMT'16 De-En	96.77	73.08
MS COCO	99.14	87.25
Flickr30K	98.63	82.17

Table 4 Accuracy of the discriminator on the development dataset.

Task	Model		eREP (↓)	DROP (↓)	BLEU (BP)	METEOR
WAT'15 Ja-En	RNMT	MLE	2.78	15.03	25.28 (1.000)	31.31
		RL-$\mathrm{D}_{\mathrm{REP}}$	**2.53** †	15.03	**25.50** (1.000)	31.19
		RL-$\mathrm{D}_{\mathrm{DROP}}$	2.67	**14.63** †	25.24 (1.000)	**31.34**
	Transformer	MLE	2.00	12.46	28.58 (1.000)	33.09
		RL-$\mathrm{D}_{\mathrm{REP}}$	**1.83** †	12.24 †	**28.93** † (1.000)	33.27 †
		RL-$\mathrm{D}_{\mathrm{DROP}}$	2.03	**11.95** †	28.75 (1.000)	**33.45** †
WMT'16 De-En	RNMT	MLE	1.09	3.87	**24.65** (1.000)	29.71
		RL-$\mathrm{D}_{\mathrm{REP}}$	**0.78**	4.07	24.60 (0.993)	29.57
		RL-$\mathrm{D}_{\mathrm{DROP}}$	0.89	**3.59** †	24.56 (1.000)	**29.87** †
	Transformer	MLE	**0.30**	3.45	27.19 (0.973)	31.64
		RL-$\mathrm{D}_{\mathrm{REP}}$	0.31	3.36 †	**27.39** † (0.973)	31.72 †
		RL-$\mathrm{D}_{\mathrm{DROP}}$	0.31	**3.17** †	27.35 (0.979)	**31.78** †

Table 5 Results on the translation tasks.

Task	Model	eREP (↓)	ROUGE$_L$	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr
MS COCO	MLE	7.42	49.80	64.71	47.09	34.41	25.69	82.77
	RL-D$_{\text{REP}}$	**6.51** †	**49.95**	**66.16** †	**48.21** †	**35.35** †	**26.54** †	**83.06**
	RL-D$_{\text{DROP}}$	7.59	49.63	64.20	46.36	33.69	25.02	82.02
Flickr30K	MLE	9.25	43.35	60.52	41.78	28.64	19.47	41.24
	RL-D$_{\text{REP}}$	**5.51** †	42.53	**62.90** †	**43.80** †	**30.29** †	**20.76**	40.75
	RL-D$_{\text{DROP}}$	6.80 †	**43.54**	61.74	43.20	29.93	20.63	**42.29**

Table 6 Results on the captioning tasks.

That is, identifying whether the dropping error occurred in a sentence would be an easy task for the discriminator; however, generating such tokens would be difficult for the generation model. We also observed that the balancing factor λ_{mixed} was preferred to be smaller for RL-D$_{\text{REP}}$ than for RL-D$_{\text{DROP}}$. This indicates that the generation model relies more on the reward from D$_{\text{REP}}$ than from D$_{\text{DROP}}$, which explains why RL-D$_{\text{REP}}$ produced more expected results than RL-D$_{\text{DROP}}$ in the captioning tasks.[9]

We show examples of the generation performance of the RNMT-based models in the WAT'15 Ja-En translation task in Table 7. In Example (A), RL-D$_{\text{REP}}$ successfully suppressed the repeated tokens that were found in MLE and generated a non-repetitive sentence. In Example (B), RL-D$_{\text{DROP}}$ satisfactorily restored the dropped tokens in MLE, and thus, generated a more informative sentence.

We report the computation time of each model on the translation tasks in Table 8. For all the experiments, we used an `Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz` and a single GPU; `NVIDIA GeForce GTX 1080 Ti`. Note that the pre-training step (MLE) required 50,000 iterations for the RNMT and 100,000 iterations for the Transformer, whereas the reinforcement learning step required 20,000 iterations for both models.

5.2 Incorporating an off-the-shelf reward function

In Section 5.1, we report that our proposed discriminator can suppress the targeted type of errors. We investigate how the performance of existing reward functions compare with our discriminators to further improve the generation performance by incorporating them. Taking GLEU (Wu et al. 2016) as an example of existing rewards, we propose the following joint reward function.

[9] For example, we considered 5.0×10^{-3} and 5.0×10^{-3} for RL-D$_{\text{REP}}$, and 0.1 and 5.0×10^{-2} for RL-D$_{\text{DROP}}$ on the COCO and Flickr30K tasks, respectively.

Example (A)	
Source	Ｉの主なものにシメチジン，ラニチジン，ファモチジンなどがあり，１日１回投与と２回投与で治癒率に差を認めない。
Reference	There are Cimetidine, Ranitidine, Famotidine, etc. in I, and differences are not recognized at therapeutic ratio in first 1 time administration and 2 time administration.
MLE	There were cimetidine, ranitidine, famotidine, etc. in the main thing of I, and the differencewas not recognized at 1 time administration and 2 time administration, and the difference was not recognized.
RL-D$_{\text{REP}}$	There were cimetidine, ranitidine, famotidine, etc. in the main thing of I, and the difference was not recognized at 1 time administration and 2 time administration.
Example (B)	
Source	その結果，電気光学特性として，Ｖ１０＝１１．８Ｖ，Ｖ９０＝１８Ｖを得た
Reference	The electro - optical property obtained was V10=11.8V and V90=18V.
MLE	As a result, V10=11.8V and V90=18V were obtained.
RL-D$_{\text{DROP}}$	As a result, V10=11.8V and V90=18V were obtained as the electro - optic characteristics.

Table 7 Generation examples of the RNMT-based models on the Ja-En translation task. The underlined tokens with solid lines represent the repeating error. The underlined tokens with a dashed line are tokens MLE failed to generate.

Task	Model	Time [h]		
		MLE	RL-D$_{\text{REP}}$	RL-D$_{\text{DROP}}$
WAT'15 Ja-En	RNMT	7.63	9.42	9.45
	Transformer	85.16	20.61	22.24
WMT'16 De-En	RNMT	9.45	10.27	10.56
	Transformer	56.13	24.14	24.41

Table 8 Computation time on the translation tasks.

$$\begin{aligned} R(s,t) =& \lambda_{\text{RL}} R'(s,t) \\ &+ (1-\lambda_{\text{RL}}) GLEU(t,r), \end{aligned} \tag{7}$$

where R' is one of our reward functions, and $GLEU$ is the GLEU score. GLEU is known to be effective in improving the BLEU score (Wu et al. 2016). $GLEU(t,r)$ calculates the minimum of the generated sentence t's n-gram precision and recall against the reference r. λ_{RL} is a hyperparameter that controls the strength of the two signals. In this section, we use RL-GLEU to refer to the model that uses only the GLEU as the reward and RL-GLEU-D$_{\text{REP}}$ (or RL-GLEU-D$_{\text{DROP}}$) to refer to the model that uses both the GLEU and its corresponding discriminator. We chose the best models based on the best BLEU scores from the development datasets.

Table 9 shows the results from the translation tasks. In the Ja-En task, RL-GLEU contributed to the improvement in the BLEU, DROP, and METEOR scores. Because the GLEU computed a recall against the reference, RL-GLEU consequently generated more informative tokens and improved those scores. Thus, RL-GLEU-D$_{\text{DROP}}$ showed less improvement on its DROP score compared to previous experiments. RL-GLEU-D$_{\text{REP}}$ further improved the eREP, particularly when RL-GLEU deteriorates the eREP score. In the De-En task, the Transformer-based RL-GLEU-D$_{\text{REP}}$ and RL-GLEU-D$_{\text{DROP}}$ showed almost the same results as RL-GLEU. This might be due to the large number of training examples; we will discuss the possible effect of the training data size in Section 5.4.

Table 10 shows the results from the image captioning tasks. In both the COCO and Flickr30K tasks, RL-GLEU significantly improved the BLEU and CIDEr scores, but generated more repetitions as indicated by the eREP scores. RL-GLEU-D$_{\text{REP}}$ achieved further improvements on the BLEU scores by suppressing the occurrence of repeating errors. RL-GLEU-D$_{\text{DROP}}$, alternatively, did not improve the performance of RL-GLEU. This may be because it is difficult to suppress dropping errors even when using both the GLEU and discriminator's signals, as discussed in Section 5.1.

Task	Model		eREP (↓)	DROP (↓)	BLEU (BP)	METEOR
WAT'15 Ja-En	RNMT	MLE	2.78	15.03	25.28 (1.000)	31.31
		RL-GLEU	2.84	13.37 †	25.73 (1.000)	**32.19** †
		RL-GLEU-D$_{\text{REP}}$	**2.65**	13.81 †	**25.83** † (1.000)	31.94 †
		RL-GLEU-D$_{\text{DROP}}$	2.96	**13.13** †	25.76 † (1.000)	32.28 †
	Transformer	MLE	2.00	12.46	28.58 (1.000)	33.09
		RL-GLEU	2.02	11.11 †	28.67 (1.000)	33.77 †
		RL-GLEU-D$_{\text{REP}}$	**1.94**	11.30 †	**28.99** (1.000)	33.75 †
		RL-GLEU-D$_{\text{DROP}}$	1.96	**11.05** †	28.92 (1.000)	**33.86** †
WMT'16 De-En	RNMT	MLE	1.09	3.87	24.65 (1.000)	29.71
		RL-GLEU	0.65 †	**2.76** †	24.57 (1.000)	30.08 †
		RL-GLEU-D$_{\text{REP}}$	**0.59** †	2.86 †	24.70 (0.998)	30.01 †
		RL-GLEU-D$_{\text{DROP}}$	0.68	2.85 †	**24.72** (1.000)	**30.11** †
	Transformer	MLE	0.30	3.45	27.19 (0.973)	31.64
		RL-GLEU	**0.29**	**3.08** †	27.13 (0.973)	**31.74** †
		RL-GLEU-D$_{\text{REP}}$	0.30	3.20 †	**27.20** (0.970)	31.69 †
		RL-GLEU-D$_{\text{DROP}}$	**0.29**	3.10	27.09 (0.974)	31.69 †

Table 9 Results of the joint models with GLEU on the translation tasks. Note that the MLE results are the same in Table 5.

Task	Model	eREP (↓)	ROUGE$_L$	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr
MS COCO	MLE	**7.42**	49.80	64.71	47.09	34.41	25.69	82.77
	RL-GLEU	10.65	51.99 †	68.83 †	52.05 †	38.28 †	28.06 †	87.87 †
	RL-GLEU-D$_{\text{REP}}$	7.96	**52.09** †	**70.31** †	**53.09** †	**39.01** †	**28.74** †	89.11 †
	RL-GLEU-D$_{\text{DROP}}$	9.04	51.99 †	69.06 †	51.97 †	38.21 †	28.21 †	**89.41** †
Flickr30K	MLE	9.25	43.35	60.52	41.78	28.64	19.47	41.24
	RL-GLEU	5.65 †	**44.08** †	63.02 †	44.41 †	30.79 †	21.19 †	**41.54**
	RL-GLEU-D$_{\text{REP}}$	5.32 †	43.97	**63.62** †	**44.86** †	**31.12** †	**21.34** †	41.27
	RL-GLEU-D$_{\text{DROP}}$	**5.05** †	43.82	63.42 †	44.50 †	30.67 †	20.95 †	41.10

Table 10 Results of the joint models with GLEU on the captioning tasks. Note that the MLE results are the same in Table 6.

5.3 Comparison with related studies

In the field of machine translation, there are two domains of research that share similarities with our research direction: (i) methods based on coverage and (ii) methods based on the generative adversarial network (GAN). The coverage-based method (Mi et al. 2016; Tu et al. 2017; Malaviya et al. 2018) uses attention history to reduce the number of repeating and dropping errors. Alternatively, the GAN-based method (Gu et al. 2018; Yang et al. 2018a) utilizes adversarial training with a discriminator to train the generation model, to generate more natural sentences. In this section, we conduct further experiments to compare the performance of these alternate methods with that of ours. In this section, we focus on the RNMT model.

5.3.1 Comparison with the coverage-based method

We first conducted experiments using the coverage-based model. We decided to use the coverage vector (Tu et al. 2017), which stores the attention history and uses it in the decoding process to more effectively utilize untranslated source words. We used neural network-based coverage, which uses RNNs to model the coverage vector, and set the coverage dimension as 10 following the procedures outlined in (Tu et al. 2017). Although Tu et al. (2017) used a gated recurrent unit as an activation function, we used LSTM, and empirically confirmed that LSTM performs well in this context. We refer to this model as CovVec.

Table 11 shows the results of this comparison. For both the two translation tasks, we can see that CovVec successfully improved the eREP and DROP scores from the MLE results, as expected. Compared to Table 5, the eREP scores were close to those of RL-D$_{\text{REP}}$ (2.53 in the Ja-En task and 0.78 in the De-En task), and the drop scores are reasonably higher than those of RL-D$_{\text{DROP}}$ (14.63 in the Ja-En task and 3.59 in the De-En task). However, the CovVec did not produce any significant improvements in the BLEU and METEOR scores in our experiments,

Task	Model	eREP (↓)	DROP (↓)	BLEU (BP)	METEOR
WAT'15 Ja-En	MLE	2.78	15.03	25.28 (1.000)	31.31
	CovVec	2.51	14.97	25.22 (1.000)	31.25
	GGD	2.88	14.98	25.41 (1.000)	31.48 †
WMT'16 De-En	MLE	1.09	3.87	24.65 (1.000)	29.71
	CovVec	0.86	3.79	24.66 (1.000)	29.58
	GGD	0.87	3.74 †	24.75 (1.000)	29.79 †

Table 11 Results of the coverage-based and GAN-based models. Note that the MLE results are the same in Table 5.

although RL-$\mathrm{D}_{\mathrm{REP}}$ in the Ja-En task achieved an improvement in BLEU. We consider that this improvement results from the following fact. By fine-tuning the parameters of MLE, our model was able to concentrate on suppressing the targeted type of errors for erroneous sentences, and continue to generate the same sentences for other, non-problematic sentences. When we checked the translations of RL-$\mathrm{D}_{\mathrm{REP}}$ and MLE, we found that RL-$\mathrm{D}_{\mathrm{REP}}$ tended to generate nearly identical translations of MLE for some source sentences, supporting our view.

5.3.2 Comparison with the GAN-based method

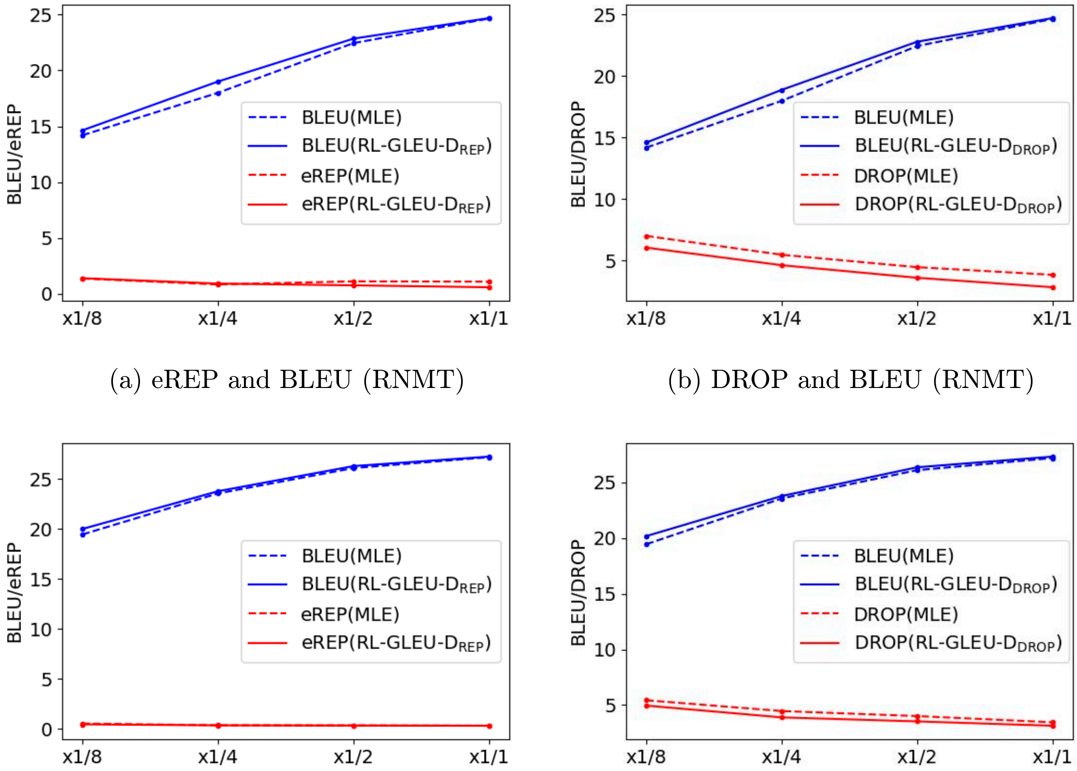
Next, we conducted experiments on the GAN-based model. We decided to use the Gumbel-Greedy decoding (GGD) method (Gu et al. 2018), which bridges the generation model and discriminator using the Gumbel-Softmax estimator (Jang et al. 2017). The hyperparameters (N_g, N_d) in (Gu et al. 2018) were set to (1, 1). We refer to this model as GGD.

The results of this comparison are shown in Table 11. For BLEU and METEOR, the GGD consistently improved both scores from the MLE results. Furthermore, the GGD had worse the eREP scores, but improved the DROP scores in the Ja-En task; conversely, in the De-En task, the GGD improved both scores. This indicates that the GGD generation model did not always learn to suppress both repeating and dropping errors. One possible reason for this is that the type of error that the discriminator handled can differ meaningfully, as the frequency of each error type included in machine-generated sentences changes depending on the task and generation model.

5.4 Varying training set size

We have confirmed that our proposed method performs effectively to suppress the targeted type of errors on the WAT'15 Ja-En translation task, as well as the COCO and Flickr30K captioning tasks, but is less effective in the WMT'16 De-En task, especially during joint training with the GLEU. One possible reason is that the size of the training dataset in the De-En translation task was sufficiently large to train the language model well and suppress the targeted errors caused by data sparseness. We hypothesize that our method would be more effective when the training dataset is small; as in such a situation, the traditional language model would not be trained well, and would produce more errors.

To verify this hypothesis, we conducted experiments by varying the training dataset size exponentially (1/1, 1/2, 1/4, 1/8) for the WMT'16 De-En translation task. For the sake of simplicity, we changed only the training dataset size, and set the same hyperparameters as those used in Section 5.2. Figure 2 shows the results of this comparison. Note that the 1/1 case is the same as that in Section 5.2. Figure 2 (a), (b), and (d) show that the results meet our expectation; namely when the size of the training dataset is smaller, our strategy is more effective with respect to the eREP, DROP, and BLEU scores. In Figure 2 (c), in contrast, RL-GLEU-D$_{\text{REP}}$ is less effective because the Transformer language model appears to be sufficiently strong to suppress most of the repetitions, even with the smallest training dataset.



(a) eREP and BLEU (RNMT)

(b) DROP and BLEU (RNMT)

(c) eREP and BLEU (Transformer)

(d) DROP and BLEU (Transformer)

Fig. 2 The results on the WMT'16 De-En translation task when varying the training size exponentially.

5.5 Sampling parameters for generating artificial negative examples

In the previous sections, we trained the discriminator with ANEs, and negative examples were created based on the parameters (i, j, k) described in Section 3.1. However, there is no guarantee that the generated errors matched the distribution nor tendency of the errors produced by the generation model in practice; thus, there is a chance that the default parameter settings for the negative examples might be inappropriate, and tuning the parameters for each model might obtain further improvement. To examine this, we conducted an additional experiment wherein we trained the discriminator, and used ANEs with the parameters sampled from the statistics of the actual generation errors.

We tested this on the WAT'15 Ja-En task using the RNMT model. First, we sampled 100 examples from the translation results of the development dataset. We then examined the phrase length, starting index of the errors, number of repetitions or droppings of the phrase, and their parts-of-speech. We found 10 repeating and 28 dropping errors. The statistics of these errors are shown in Table 12. We then trained the discriminator with ANEs by sampling the phrase length i in Section 3.1 from the statistics. For the repeating errors, we also sampled the number of repetitions (k in Section 3.1) from the statistics. For the starting index j in Section 3.1, we continued to sample the value uniformly, as there was no tendency or regularity for the starting index of the errors. We present the experimental results of the generation models trained with the discriminators in Table 13. We add "sampling" to the model names to indicate the enhanced models introduced in this section. Compared with the results of RL-D$_{\text{REP}}$ and RL-D$_{\text{DROP}}$ from Table 5, training the discriminator with ANEs specially tuned for the model has little effect on the overall performance of the generation model. One possible reason for this is that the discriminator trained with the negative examples without sampling parameters already had the ability to capture the errors of the generation models, and thus, performed sufficiently well to discriminate the errors during the reinforcement learning step.

6 Related work

Training neural text generation models with a discriminator has recently been studied in GAN-based methods (Dai et al. 2017; Gu et al. 2018; Yang et al. 2018b). These studies utilized the generative adversarial framework (Goodfellow et al. 2014; Arjovsky et al. 2017) to train sentence generation models to generate more natural, human-like sentences. Although our framework can also be considered as such a generator-discriminator framework, there are two points that differentiate our framework from the GAN-based methods. First, our generator can focus on

Phrase length	Starting index	# of repetitions or droppings	Part-of-speech
Repeating error			
2(4), 3(3), 1(2), 7(1)	5(1), 8(1), 12(1), 14(1), 17(1), 19(1), 22(1), 23(1), 24(1), 27(1), 33(1), 34(1)	1(9), 3(1)	Noun(8), Adverb(1), Verb(1)
Dropping error			
1(13), 3(5), 2(4), 4(4), 9(1)	1(4), 5(3), 10(3), 19(3), 24(2), 26(2), 4(1), 8(1), 12(1), 13(1), 14(1), 15(1), 17(1), 18(1), 28(1), 31(1), 32(1)	1(26), 2(1)	Noun(15), Adjective(5), Adverb(3), Conjunction(3), Numeral(2)

Table 12 Statistics of errors sampled from 100 translation results of the RNMT model on the WAT'15 Ja-En task. The number in the parentheses represents the frequency.

Task	Model	eREP (↓)	DROP (↓)	BLEU	(BP)	METEOR
WAT'15 Ja-En	RL-D$_{\text{REP}}$	2.53	15.03	25.50	(1.000)	31.19
	RL-D$_{\text{REP}}$+ sampling	2.46	15.75	25.34	(1.000)	30.92
	RL-D$_{\text{DROP}}$	2.67	14.63	25.24	(1.000)	31.34
	RL-D$_{\text{DROP}}$+ sampling	2.65	14.65	25.16	(1.000)	31.31

Table 13 Results

suppressing a targeted type of error, as the discriminator learns to discriminate that error type from the references with artificially generated erroneous sentences. The other is that our ANEs definitely contain errors, whereas the machine-generated sentences in the GAN-based methods can include correct sentences, as they are sampled from the generation model.

Several studies have used negative examples to train sentence generation model. Focusing on the use of repetitions, Welleck et al. (2020) proposed the technique of unlikelihood training, wherein the training objective was to decrease the probability of generating the token that had appeared in the previous context. He and Glass (2020) collected negative examples from the model generations, and trained the model to no longer generate them. One advantage of our method is that our discriminator is trained independent of the sentence generation model; consequently, our method does not require any extra steps to collect negative examples from the sentence generation model.

Coverage-based methods in machine translation have also aimed to suppress the occurrence of repeating and dropping errors by focusing on the coverage information regarding the source words (Mi et al. 2016; Tu et al. 2017; Malaviya et al. 2018). In abstractive summarization, Suzuki and

Nagata (2017) attempted to address repeating errors by controlling the output words with the estimated upper-bound frequency of words to be generated. Although our method is not limited to these two error types, there are two obstacles to overcome for addressing different types of errors than the ones assessed here. First is the problem of classifying the error type, which can be as difficult as the actual generation of outputs containing the various types of errors with a certain frequency. Hence, detecting and classifying all error types would seem to be a difficult task that is likely to fail. The other main obstacle is designing the corresponding ANEs for these other error types. This would also be difficult if the errors are difficult to produce based on the reference sentences that we relied on in this study.

Automatic post-editing aims to edit machine-translated sentences to fix errors, writing styles, and so on (Freitag et al. 2019; Gu et al. 2019). The task is similar to ours, in that it considers how model-generated sentences should be improved. However, there are two significant differences. First, researchers studying automatic post-editing typically use a machine translation model and post-editing model independently, whereas we consistently trained one model to generate fewer erroneous sentences. Second, our model explicitly focused on one specific generation error type, which was not feasible in the previous work.

7 Conclusion

We propose a reinforcement learning-based method that directly suppresses a targeted type of error, unlike conventional methods. Our method uses a discriminator that captures a specific type of error, and this discriminator is trained with artificially generated negative examples. We empirically demonstrate that our approach can effectively reduce the number of repeating and dropping errors. In future work, it will be of interest to explore the application of our method to source-side, unannotated data.

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). "Wasserstein Generative Adversarial Networks." In *International Conference on Machine Learning*, pp. 214–223.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). "Neural Machine Translation by Jointly Learning to Align and Translate." In *3rd International Conference on Learning Representations.*

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollár, P., and Zitnick, C. L. (2015). "Microsoft Coco Captions: Data Collection and Evaluation Server." *arXiv preprint* *arXiv:1504.00325v2*.

Dai, B., Fidler, S., Urtasun, R., and Lin, D. (2017). "Towards Diverse and Natural Image Descriptions via a Conditional Gan." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2970–2979.

Freitag, M., Caswell, I., and Roy, S. (2019). "APE at Scale and Its Implications on MT Evaluation Biases." In *Proceedings of the 4th Conference on Machine Translation (Volume 1: Research Papers)*, pp. 34–44, Florence, Italy. Association for Computational Linguistics.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). "Generative Adversarial Nets." In *Advances in Neural Information Processing Systems*, pp. 2672–2680.

Gu, J., Im, D. J., and Li, V. O. (2018). "Neural Machine Translation with Gumbel-Greedy Decoding." In *The Association for the Advancement of Artificial Intelligence.*

Gu, J., Wang, C., and Zhao, J. (2019). "Levenshtein Transformer." In *Advances in Neural Information Processing Systems*, pp. 11179–11189.

Hashimoto, K. and Tsuruoka, Y. (2019). "Accelerated Reinforcement Learning for Sentence Generation by Vocabulary Prediction." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3115–3125, Minneapolis, Minnesota. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep Residual Learning for Image Recognition." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

He, T. and Glass, J. (2020). "Negative Training for Neural Dialogue Response Generation." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2044–2058. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). "Long Short-term Memory." *Neural Computation*, **9** (8), pp. 1735–1780.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). "The Curious Case of Neural Text Degeneration." In *International Conference on Learning Representations.*

Jang, E., Gu, S., and Poole, B. (2017). "Categorical Reparameterization with Gumbel-Softmax." In *5th International Conference on Learning Representations.*

Karpathy, A. and Fei-Fei, L. (2015). "Deep Visual-semantic Alignments for Generating Image Descriptions." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137.

Kingma, D. P. and Ba, J. (2014). “Adam: A Method for Stochastic Optimization.” In *International Conference on Learning Representations.*

Koehn, P. (2004). “Statistical significance tests for machine translation evaluation.” In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388–395.

Kudo, T. and Richardson, J. (2018). “SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing.” In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). “Microsoft Coco: Common objects in Context.” In *European Conference on Computer Vision*, pp. 740–755. Springer.

Loshchilov, I. and Hutter, F. (2019). “Decoupled Weight Decay Regularization.” In *International Conference on Learning Representations.*

Malaviya, C., Ferreira, P., and Martins, A. F. T. (2018). “Sparse and Constrained Attention for Neural Machine Translation.” In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 2*, pp. 370–376.

Mi, H., Sankaran, B., Wang, Z., and Ittycheriah, A. (2016). “Coverage Embedding Models for Neural Machine Translation.” In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 955–960, Austin, Texas.

Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). “ASPEC: Asian Scientific Paper Excerpt Corpus.” In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). “BLEU: A Method for Automatic Evaluation of Machine Translation.” In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). “On the Difficulty of Training Recurrent Neural Networks.” In *International Conference on Machine Learning*, pp. 1310–1318.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). “Flickr30k Entities: Collecting Region-to-phrase Correspondences for Richer Image-to-sentence Models.” In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2641–2649.

Ranzato, M. A., Chopra, S., Auli, M., and Zaremba, W. (2016). “Sequence Level Training with

Recurrent Neural Networks." *In 6th International Conference on Learning Representations.*

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). "Self-critical Sequence Training for Image Captioning." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). "Imagenet Large Scale Visual Recognition Challenge." *International Journal of Computer Vision*, **115** (3), pp. 211–252.

Sennrich, R., Haddow, B., and Birch, A. (2016). "Neural Machine Translation of Rare Words with Subword Units." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). "Sequence to Sequence Learning with Neural Networks." In *Advances in Neural Information Processing Systems*, pp. 3104–3112.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.

Suzuki, J. and Nagata, M. (2017). "Cutting-off Redundant Repeating Generations for Neural Abstractive Summarization." In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 291–297. Association for Computational Linguistics.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). "Rethinking the Inception Architecture for Computer Vision." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Tu, Z., Liu, Y., Lu, Z., Liu, X., and Li, H. (2017). "Context Gates for Neural Machine Translation." *Transactions of the Association for Computational Linguistics*, **5**, pp. 87–99.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). "Attention is All you Need." In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc.

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

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). "Show and Tell: A Neural Image Caption Generator." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. (2020). "Neural Text Generation with Unlikelihood Training." In *8th International Conference on Learning Rep-*

resentations.

Williams, R. J. (1992). "Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning." *Machine Learning*, **8** (3-4), pp. 229–256.

Williams, R. J. and Zipser, D. (1989). "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks." *Neural Computation*, **1** (2), pp. 270–280.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." *arXiv preprint arXiv:1609.08144v2.*

Yang, Z., Chen, W., Wang, F., and Xu, B. (2018a). "Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets." In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pp. 1346–1355.

Yang, Z., Hu, Z., Dyer, C., Xing, E. P., and Berg-Kirkpatrick, T. (2018b). "Unsupervised Text Style Transfer using Language Models as Discriminators." In *Advances in Neural Information Processing Systems*, pp. 7287–7298.

Zhang, X. and Lapata, M. (2017). "Sentence Simplification with Deep Reinforcement Learning." In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 584–594. Association for Computational Linguistics.

Keisuke Shirai: Keisuke Shirai received his B.S. from Ehime University in 2017 and M.S. from Kyoto University in 2020. He is currently a Ph.D student in the Graduate School of Informatics at Kyoto University. His research interests include natural language processing, reinforcement learning, and symbol grounding.

Kazuma Hashimoto: Kazuma Hashimoto received his B.S., M.S., and Ph.D from the University of Tokyo in 2013, 2015, and 2018, respectively. He is currently a researcher at Salesforce Research. His research interests include natural language processing both from academic and applied sides.

Akiko Eriguchi: Akiko Eriguchi received her B.S. and M.S. from Ochanomizu University in 2013 and 2015, respectively, and Ph.D from the University of Tokyo in 2018. She is currently a research scientist at Microsoft Corporation. Her research interests include natural language processing and machine learning. She is a member of the Japan Association for Natural Language

Processing and the Association for Computational Linguistics.

Takashi Ninomiya: Takashi Ninomiya received his B.S., M.S., and Ph.D from the University of Tokyo in 1996, 1998, and 2001, respectively. He is currently a professor at the Graduate School of Science and Engineering, Ehime University. His research interests include natural language processing and computational linguistics. He is a member of the Japan Association for Natural Language Processing, the Information Processing Society of Japan, the Japanese Society of Artificial Intelligence, the Institute of Electronics, Information and Communication Engineers, the Asia-Pacific Association for Machine Translation, and the Database Society of Japan.

Shinsuke Mori: Shinsuke Mori received his B.S., M.S., and Ph.D from Kyoto University in 1993, 1995, and 1998, respectively. After joining the Tokyo Research Laboratory of International Business Machines in 1998, he studied language modeling and its application to speech recognition and language processing. He is currently a professor at the Academic Center for Computing and Media Studies, Kyoto University. His research interests include natural language processing, spoken language processing, human-computer interaction, and multimedia integration. He is a member of the Japan Association for Natural Language Processing, and the Information Processing Society of Japan, and the Database Society of Japan.

(Received December 28, 2020)
(Revised April 7, 2021)
(Accepted May 11, 2021)