Rethinking Perturbations in Encoder-Decoders for Fast Training

Sho Takase
Tokyo Institute of Technology
sho.takase@nlp.c.titech.ac.jp

Shun Kiyono
RIKEN / Tohoku University
shun.kiyono@riken.jp

Abstract

We often use perturbations to regularize neural models. For neural encoder-decoders, previous studies applied the scheduled sampling (Bengio et al., 2015) and adversarial perturbations (Sato et al., 2019) as perturbations but these methods require considerable computational time. Thus, this study addresses the question of whether these approaches are efficient enough for training time. We compare several perturbations in sequence-to-sequence problems with respect to computational time. Experimental results show that the simple techniques such as word dropout (Gal and Ghahramani, 2016) and random replacement of input tokens achieve comparable (or better) scores to the recently proposed perturbations, even though these simple methods are faster. Our code is publicly available at https://github.com/takase/rethink_perturbations.

1 Introduction

Recent advances in neural encoder-decoders have driven tremendous success for sequence-to-sequence problems including machine translation (Sutskever et al., 2014), summarization (Rush et al., 2015), and grammatical error correction (GEC) (Ji et al., 2017). Since neural models can be too powerful, previous studies have proposed various regularization methods to avoid over-fitting.

To regularize neural models, we often apply a perturbation (Goodfellow et al., 2015; Miyato et al., 2017), which is a small difference from a correct input. During the training process, we force the model to output the correct labels for both perturbed inputs and unmodified inputs. In sequence-to-sequence problems, existing studies regard the following as perturbed inputs: (1) sequences containing tokens replaced from correct ones (Bengio et al., 2015; Cheng et al., 2019), (2) embeddings injected small differences (Sato et al., 2019). For example, Bengio et al. (2015) proposed the scheduled sampling that samples a token from the output probability distribution of a decoder and uses it as a perturbed input for the decoder. Sato et al. (2019) applied an adversarial perturbation, which significantly increases the loss value of a model, to the embedding spaces of neural encoder-decoders.

Those studies reported that their methods are effective to construct robust encoder-decoders. However, their methods are much slower than the training without using such perturbations because they require at least one forward computation to obtain the perturbation. In fact, we need to run the decoder the same times as the required number of perturbations in the scheduled sampling (Bengio et al., 2015). For adversarial perturbations (Sato et al., 2019), we have to compute the backpropagation in addition to forward computation because we use gradients to obtain perturbations.

Those properties seriously affect the training budget. For example, it costs approximately 1,800 USD for each run when we train Transformer (big) with adversarial perturbations (Sato et al., 2019) on the widely used WMT English-German training set in AWS EC2[1]. Most studies conduct multiple runs for the hyper-parameter search and/or model ensemble to achieve better performance (Barrault et al., 2019), which incurs a tremendous amount of training budget for using such perturbations. Strubell et al. (2019) and Schwartz et al. (2019) indicated that recent neural approaches increase computational costs substantially, and they encouraged exploring a cost-efficient method. For instance, Li et al. (2020) explored a training strategy to obtain the best model in a given training time. However, previous studies have paid little attention to the costs of computing perturbations.

Thus, we rethink a time efficient perturbation method. In other words, we address the question whether perturbations proposed by recent studies as effective methods are time efficient. We compare

[1]We assume that we use on-demand instances having 8 V100 GPUs.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5767–5780
June 6–11, 2021.

several perturbation methods for neural encoder-decoders in terms of computational time. We introduce light computation methods such as word dropout (Gal and Ghahramani, 2016) and using randomly sampled tokens as perturbed inputs. These methods are sometimes regarded as baseline methods (Bengio et al., 2015), but experiments on translation datasets indicate that these simple methods surprisingly achieve comparable scores to those of previous effective perturbations (Bengio et al., 2015; Sato et al., 2019) in a shorter training time. Moreover, we indicate that these simple methods are also effective for other sequence-to-sequence problems: GEC and summarization.

2 Definition of Encoder-Decoder

In this paper, we address sequence-to-sequence problems such as machine translation with neural encoder-decoders, and herein we provide a definition of encoder-decoders.

In sequence-to-sequence problems, neural encoder-decoders generate a sequence corresponding to an input sequence. Let $\boldsymbol{x}_{1:I}$ and $\boldsymbol{y}_{1:J}$ be input and output token sequences whose lengths are I and J, respectively: $\boldsymbol{x}_{1:I} = x_1, ..., x_I$ and $\boldsymbol{y}_{1:J} = y_1, ..., y_J$. Neural encoder-decoders compute the following conditional probability:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J+1} p(y_j|\boldsymbol{y}_{0:j-1}, \boldsymbol{X}), \quad (1)$$

where y_0 and y_{J+1} are special tokens representing beginning-of-sentence (BOS) and end-of-sentence (EOS) respectively, $\boldsymbol{X} = \boldsymbol{x}_{1:I}$, and $\boldsymbol{Y} = \boldsymbol{y}_{1:J+1}$.

In the training phase, we optimize the parameters $\boldsymbol{\theta}$ to minimize the negative log-likelihood in the training data. Let $\mathcal{D}$ be the training data consisting of a set of pairs of $\boldsymbol{X}_n$ and $\boldsymbol{Y}_n$: $\mathcal{D} = \{(\boldsymbol{X}_n, \boldsymbol{Y}_n)\}_{n=1}^{|\mathcal{D}|}$. We minimize the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \log p(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\theta}). \quad (2)$$

3 Definition of Perturbations

This section briefly describes perturbations used in this study. This study focuses on three types of perturbations: *word replacement*, *word dropout*, and *adversarial perturbations*. Figure 1 shows perturbations used in this study. As shown in this figure, we can use all types of perturbations in the same time because perturbations are orthogonal to each other. In fact, we combine word replacement with word dropout in our experiments.

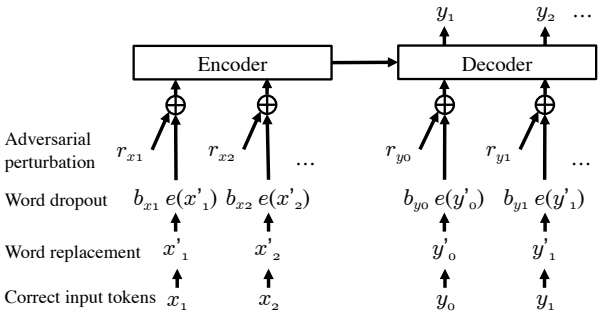


Figure 1: Overview of perturbations used in this study. We can combine perturbations as shown in this figure because each type of perturbation is orthogonal.

3.1 Word Replacement: REP

For any approach that uses a sampled token instead of a correct token, such as the scheduled sampling (Bengio et al., 2015), we refer to this as a word replacement approach. In this approach, we construct a new sequence whose tokens are randomly replaced with sampled tokens. For the construction from the sequence $\boldsymbol{X}$, we sample $\hat{x}_i$ from a distribution Q_{x_i} and use it for the new sequence $\boldsymbol{X}'$ with the probability α:

$$\hat{x}_i \sim Q_{x_i}, \quad (3)$$

$$x'_i = \begin{cases} x_i & \text{with probability } \alpha \\ \hat{x}_i & \text{with probability } 1-\alpha. \end{cases} \quad (4)$$

We construct $\boldsymbol{Y}'$ from the sequence $\boldsymbol{Y}$ in the same manner.

Bengio et al. (2015) used a curriculum learning strategy to adjust α, and thus proposed several functions to decrease α based on the training step. Their strategy uses correct tokens frequently at the beginning of training, whereas it favors sampled tokens frequently at the end of training. We also adjust α with their use of the inverse sigmoid decay:

$$\alpha_t = \max\left(q, \frac{k}{k + \exp(\frac{t}{k})}\right) \quad (5)$$

where q and k are hyper-parameters. In short, α_t decreases to q from 1, depending on the training step t. We use α_t as α at t.

For Q_{x_i}, we prepare three types of distributions: *conditional probability*, *uniform*, and *similarity*.

Conditional Probability: REP(SS) Bengio et al. (2015) proposed the scheduled sampling which uses predicted tokens during training to address the gap between training and inference. Formally, the scheduled sampling uses the following conditional probability as Q_{y_i}:

$$p(\hat{y}_i|\boldsymbol{y}'_{0:i-1}, \boldsymbol{X}). \quad (6)$$

Since the scheduled sampling is the method to compute the perturbation for the decoder side only, it uses the correct sequence as the input of the encoder side. In other words, the scheduled sampling does not provide any function for Q_{x_i}.

The original scheduled sampling repeats the decoding for each of the tokens on the decoder side, and thus, requires computational time in proportion to the length of the decoder-side input sequence. To address this issue, Duckworth et al. (2019) proposed a more time efficient method: parallel scheduled sampling which computes output probability distributions corresponding to each position simultaneously. In this study, we use parallel scheduled sampling instead of the original method.

Uniform: REP(UNI) The scheduled sampling is slow even if we use parallel scheduled sampling because it requires decoding at least once to compute Equation (6). Thus, we introduce two faster methods to explore effective perturbations from the perspective of computational time. In *uniform*, we use the uniform distributions on each vocabulary as Q_{x_i} and Q_{y_i}, respectively. For example, we randomly pick up a token from the source-side vocabulary and use the token as $\hat{x}_i$ in Equation (4) to construct the source-side perturbed input. This method is used as the baseline in the previous study (Bengio et al., 2015).

Similarity: REP(SIM) We also explore more sophisticated way than the uniform distribution. We assume that the conditional probability of Equation (6) assigns high probabilities to tokens that are similar to the correct input token. Based on this assumption, we construct a distribution that enables us to sample similar tokens frequently. Let $\mathcal{V}_x$ be the source-side vocabulary, $\boldsymbol{E}_x \in \mathbb{R}^{|\mathcal{V}_x| \times d_x}$ be the d_x dimensional embedding matrix, and $\boldsymbol{e}(x_i)$ be the function returning the embedding of x_i. We use the following probability distribution as Q_{x_i}:

$$\mathrm{softmax}(\boldsymbol{E}_x \boldsymbol{e}(x_i)), \quad (7)$$

where $\mathrm{softmax}(.)$ is the softmax function. Thus, Equation (7) assigns high probabilities to tokens whose embeddings are similar to $\boldsymbol{e}(x_i)$. In other words, Equation (7) is the similarity against x_i without considering any context. We compute the probability distribution for the target side by using $\boldsymbol{e}(y_i)$ in the same manner.

3.2 Word Dropout: WDROP

We apply the word dropout technique to compute the perturbed input. Word dropout randomly uses the zero vector instead of the embedding $\boldsymbol{e}(x_i)$ for the input token x_i (Gal and Ghahramani, 2016):

$$b_{x_i} \sim \mathrm{Bernoulli}(\beta), \quad (8)$$

$$\mathrm{WDrop}(x_i, b_{x_i}) = b_{x_i} \boldsymbol{e}(x_i), \quad (9)$$

where $\mathrm{Bernoulli}(\beta)$ returns 1 with the probability β and 0 otherwise. Thus, $\mathrm{WDrop}(x_i, b_{x_i})$ returns $\boldsymbol{e}(x_i)$ with the probability β and the zero vector otherwise. We apply Equation (9) to each token in the input sequence. Then, we use the results as the perturbed input.

3.3 Adversarial Perturbation: ADV

Miyato et al. (2017) proposed a method to compute adversarial perturbations in the embedding space. Their method adds adversarial perturbations to input embeddings instead of replacing correct input tokens with others. Sato et al. (2019) applied this approach to neural encoder-decoders and reported its effectiveness. Thus, this study follows the methods used in Sato et al. (2019).

The method seeks the adversarial perturbation, which seriously damages the loss value, based on the gradient of the loss function $\mathcal{L}(\boldsymbol{\theta})$. Then, we add the adversarial perturbation to the input token embedding. Let $\boldsymbol{r}_{x_i} \in \mathbb{R}^{d_x}$ be the adversarial perturbation vector for the input token x_i. We obtain the perturbed input embedding $\boldsymbol{e}'(x_i)$ with the following equations:

$$\boldsymbol{e}'(x_i) = \boldsymbol{e}(x_i) + \boldsymbol{r}_{x_i}, \quad (10)$$

$$\boldsymbol{r}_{x_i} = \epsilon \frac{\boldsymbol{c}_{x_i}}{||\boldsymbol{c}_{x_i}||}, \quad (11)$$

$$\boldsymbol{c}_{x_i} = \nabla_{\boldsymbol{e}(x_i)} \mathcal{L}(\boldsymbol{\theta}), \quad (12)$$

where ϵ is a hyper-parameter to control the norm of the adversarial perturbation. We apply the above equations to all tokens in the input sequence.

3.4 Training

In the training using word replacement and/or word dropout perturbations, we search the parameters

predicting the correct output sequence from the perturbed input. For example, in the word replacement approach, we minimize the following negative log-likelihood:

$$\mathcal{L}'(\boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}} \log p(\boldsymbol{Y}|\boldsymbol{X}', \boldsymbol{Y}'; \boldsymbol{\theta}),$$
$$= -\frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}}\sum_{j=1}^{J+1} \log p(y_j|\boldsymbol{y}'_{0:j-1}, \boldsymbol{X}'; \boldsymbol{\theta}). \quad (13)$$

Virtual Adversarial Training When we use adversarial perturbations, we train parameters of the neural encoder-decoder to minimize both Equation (2) and a loss function $\mathcal{A}(\boldsymbol{\theta})$ composed of perturbed inputs:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda\mathcal{A}(\boldsymbol{\theta}), \quad (14)$$

where λ is a hyper-parameter to control the balance of two loss functions. This calculation seems to be reasonably time efficient because adversarial perturbations require computing Equation (2).

Sato et al. (2019) used the virtual adversarial training originally proposed in Miyato et al. (2016) as a loss function for perturbed inputs. In the virtual adversarial training, we regard the output probability distributions given the correct input sequence as positive examples:

$$\mathcal{A}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}} \mathrm{KL}\left(p(\cdot|\boldsymbol{X}; \boldsymbol{\theta})||p(\cdot|\boldsymbol{X}, \boldsymbol{r}; \boldsymbol{\theta})\right), \quad (15)$$

where $\boldsymbol{r}$ represents a concatenated vector of adversarial perturbations for each input token, and $\mathrm{KL}(\cdot||\cdot)$ denotes the Kullback–Leibler divergence.

4 Experiments on Machine Translation

To obtain findings on sequence-to-sequence problems, we conduct experiments on various situations: different numbers of training data and multiple tasks. We mainly focus on translation datasets because machine translation is a typical sequence-to-sequence problem. We regard the widely used WMT English-German dataset as a standard setting. In addition, we vary the number of training data in machine translation: high resource in Section 4.2 and low resource in Section 4.3. Table 1 summarizes the number of training data in each configuration. Moreover, we conduct experiments on other sequence-to-sequence problems: grammatical error correction (GEC) in Section 5 and summarization in Appendix A to confirm whether the findings from machine translation are applicable to other tasks.

Setting	Genuine	Synthetic
Standard	4.5M	-
High Resource	4.5M	20.0M
Low Resource	160K	-

Table 1: Sizes of training datasets on our machine translation experiments.

4.1 Standard Setting

Datasets We used the WMT 2016 English-German training set, which contains 4.5M sentence pairs, in the same as Ott et al. (2018), and followed their pre-processing. We used newstest2013 as a validation set, and newstest2010-2012, and 2014-2016 as test sets. We measured case-sensitive detokenized BLEU with SacreBLEU (Post, 2018)[2].

Methods We used Transformer (Vaswani et al., 2017) as a base neural encoder-decoder model because it is known as a strong neural encoder-decoder model. We used two parameter sizes: base and big settings in Vaswani et al. (2017).

We applied perturbations described in Section 3 for comparison. For parallel scheduled sampling (Duckworth et al., 2019), we can compute output probability distributions multiple times but we used the first decoding result only because it is the fastest approach. We set $q = 0.9$, $k = 1000$, and $\beta = 0.9$. For ADV, we used the same hyper-parameters as in Sato et al. (2019). Our implementation is based on `fairseq`[3] (Ott et al., 2019). We trained each model for a total of 50,000 steps.

Preliminary: To which sides do we apply perturbations? As described, perturbations based on REP(SS) can be applied to the decoder side only. Sato et al. (2019) reported their method was the most effective when they applied their ADV to both encoder and decoder sides. However, we do not have evidence for suitable sides in applying other perturbations. Thus, we applied REP(UNI),

[2] As reported in Ott et al. (2018), the BLEU score from SacreBLEU is often lower than the score from `multi-bleu.perl` but SacreBLEU is suitable for scoring WMT datasets (Post, 2018).

[3] https://github.com/pytorch/fairseq

Method	Position	2010	2011	2012	2013	2014	2015	2016	Average
Transformer (base)									
w/o perturbation	-	24.27	22.06	22.43	26.11	27.13	29.70	34.40	26.59
REP(UNI)	enc	24.26	21.95	22.33	25.76	26.70	29.08	**34.61**	26.38
	dec	24.27	21.99	22.29	**26.31**	**27.28**	**29.74**	**34.42**	**26.61**
	both	**24.30**	**22.20**	22.43	26.06	26.82	29.42	34.13	26.48
REP(SIM)	enc	24.12	22.02	22.14	**26.21**	27.01	29.33	**34.56**	26.48
	dec	**24.32**	21.96	**22.55**	**26.36**	**27.23**	**29.86**	34.33	**26.66**
	both	23.94	21.85	22.29	25.84	26.61	29.50	34.20	26.32
WDROP	enc	**24.31**	**22.12**	**22.45**	**26.20**	27.09	**29.95**	**34.58**	**26.67**
	dec	23.96	**22.08**	22.22	**26.36**	27.08	**29.91**	33.98	26.51
	both	**24.33**	**22.14**	22.35	26.10	26.82	29.51	**34.51**	26.54
Transformer (big)									
w/o perturbation	-	24.22	22.11	22.69	26.60	28.46	30.50	33.58	26.88
REP(UNI)	enc	**24.79**	**22.49**	**23.10**	**27.07**	28.39	**30.52**	**34.51**	**27.27**
	dec	**24.33**	**22.34**	22.63	**26.93**	28.22	30.36	33.41	**26.89**
	both	**24.75**	**22.68**	**23.32**	**27.01**	**28.89**	**31.38**	**34.94**	**27.57**
REP(SIM)	enc	**24.68**	**22.91**	**23.13**	**27.03**	28.25	**30.81**	**34.40**	**27.32**
	dec	**24.51**	**22.22**	**22.83**	26.46	**28.64**	**30.68**	33.58	**26.99**
	both	**24.77**	**22.50**	**23.10**	**26.91**	**28.98**	**31.03**	**34.29**	**27.37**
WDROP	enc	**24.60**	**22.32**	**23.27**	**27.07**	28.40	**31.00**	**34.61**	**27.32**
	dec	**24.53**	**22.33**	**22.75**	**27.00**	**28.56**	**30.58**	33.20	**26.99**
	both	**24.92**	**22.71**	**23.40**	**27.11**	**28.73**	**30.99**	**34.80**	**27.52**
REP(UNI)+WDROP	both	**24.82**	**22.82**	**23.38**	**27.30**	**28.56**	**30.65**	**35.02**	**27.51**
REP(SIM)+WDROP	both	**24.83**	**22.95**	**23.40**	**27.23**	**28.65**	**30.88**	**35.05**	**27.57**

Table 2: BLEU scores on newstest2010-2016 and averaged scores. Bolds are better scores than w/o perturbations.

REP(SIM), and WDROP to the encoder side, decoder side, and both as preliminary experiments.

Table 2 shows BLEU scores on newstest2010-2016 and averaged scores when we varied the position of the perturbations. In this table, we indicate better scores than the original Transformer (Vaswani et al., 2017) (w/o perturbation) in bold. This table shows that it is better to apply word replacement (REP(UNI) and REP(SIM)) to the decoder side in Transformer (base). For WDROP, applying the encoder side is slightly better than other positions in Transformer (base). In contrast, applying perturbations to both sides achieved the best averaged BLEU scores for all methods in Transformer (big). These results imply that it is better to apply to word replacement and/or word dropout to both encoder and decoder sides if we prepare enough parameters for neural encoder-decoders. Based on these results, we select methods to compare against scheduled sampling (REP(SS)) and adversarial perturbations (ADV).

Table 2 also shows the results when we combined each word replacement with word dropout (REP(UNI)+WDROP and REP(SIM)+WDROP). REP(SIM)+WDROP slightly outperformed the separated settings.

Results We compare each perturbation in view of computational time. Table 3 shows BLEU scores of each method and computational speeds[4] based on Transformer (base) without any perturbations, i.e., larger is faster. In this table, we indicate the best score of each column for Transformer (base) and (big) settings in bold. This table indicates that Transformer without perturbations achieved a comparable score to previous studies (Vaswani et al., 2017; Ott et al., 2018) on newstest2014 in base and big settings. Thus, we consider that our trained Transformer models (w/o perturbation) can be regarded as strong baselines. This table shows that ADV achieved the best averaged score in Transformer (base), but this method required twice as much training time as the original Transformer (base). In contrast, REP(SIM) and WDROP achieved comparable scores to ADV although they slightly affected the computational time. REP(UNI) also achieved a slightly better averaged score than the original Transformer (base).

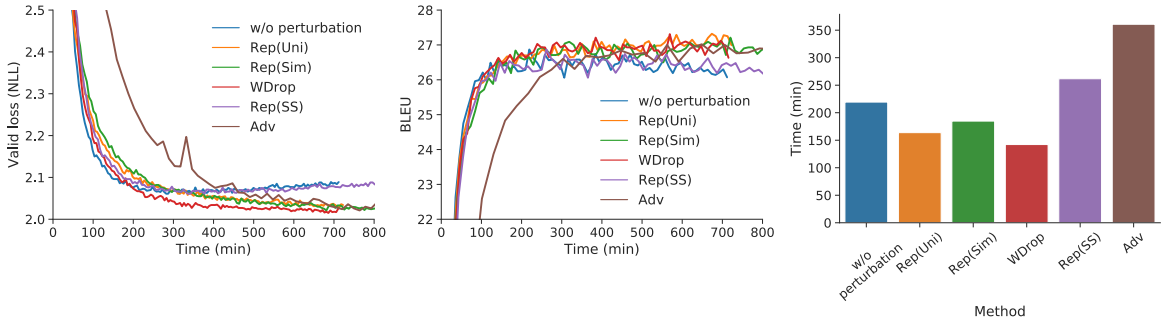
In the Transformer (big) setting, all perturbations surpassed the performance of w/o perturbation in the averaged score. REP(SS) and ADV improved the performance, but other methods outperformed these two methods with a small training time. Moreover, REP(UNI) and REP(SIM)+WDROP achieved the best averaged score.

Figure 2 illustrates the negative log-likelihood

[4]We regard processed tokens per second as the computational speed of each method.

Method	Position	2010	2011	2012	2013	2014	2015	2016	Average	Speed
Transformer (base)										
w/o perturbation	-	24.27	22.06	22.43	26.11	27.13	29.70	34.40	26.59	×1.00
REP(UNI)	dec	24.27	21.99	22.29	26.31	**27.28**	29.74	34.42	26.61	×0.99
REP(SIM)	dec	24.32	21.96	22.55	**26.36**	27.23	29.86	34.33	26.66	×0.95
WDROP	enc	24.31	22.12	22.45	26.20	27.09	**29.95**	34.58	26.67	×1.00
REP(SS)	dec	24.18	22.03	22.38	26.04	27.15	29.77	34.24	26.54	×0.88
ADV	both	**24.34**	**22.19**	**22.58**	26.19	27.10	29.78	**34.89**	**26.72**	×0.44
Transformer (big)										
w/o perturbation	-	24.22	22.11	22.69	26.60	28.46	30.50	33.58	26.88	×0.60
REP(UNI)	both	24.75	22.68	23.32	27.01	28.89	**31.38**	34.94	**27.57**	×0.60
REP(SIM)	both	24.77	22.50	23.10	26.91	**28.98**	31.03	34.29	27.37	×0.55
WDROP	both	**24.92**	22.71	**23.40**	27.11	28.73	30.99	34.80	27.52	×0.60
REP(UNI)+WDROP	both	24.82	22.82	23.38	**27.30**	28.56	30.65	35.02	27.51	×0.60
REP(SIM)+WDROP	both	24.83	**22.95**	**23.40**	27.23	28.65	30.88	**35.05**	**27.57**	×0.55
REP(SS)	dec	24.44	21.97	22.74	26.77	28.44	30.83	33.71	26.99	×0.52
ADV	both	24.71	22.60	23.23	26.98	28.97	30.49	34.40	27.34	×0.20

Table 3: BLEU scores on newstest2010-2016, averaged scores, and computational speeds based on Transformer (base) w/o perturbation. Scores in bold denote the best result for each set for Transformer (base) and (big) settings.



(a) Valid NLL (b) BLEU score (c) Time to achieve BLEU score of Transformer w/o perturbation

Figure 2: Negative log-likelihood (NLL) values, BLEU scores of each method, and time to achieve BLEU score of Transformer w/o perturbation on validation set (newstest2013).

values and BLEU scores on the validation set for each training time when we applied each perturbation to Transformer (big). In addition, Figure 2 (c) shows the time required to achieve the BLEU score of Transformer w/o perturbation on the validation set (26.60, as described in Table 3). These figures show that ADV requires twice as much time or more relative to other methods to achieve performance comparable to others. In NLL curves, REP(UNI), REP(SIM), and WDROP achieved better values than those of Transformer w/o perturbation in the early stage. In addition, WDROP was the fastest to achieve better NLL value. Figure 2 (c) indicates that REP(UNI), REP(SIM), and WDROP achieved 26.60 BLEU score with smaller training time than that of Transformer w/o perturbation.

These results indicate that we can quickly improve the performance of Transformer with REP(UNI), REP(SIM), and WDROP. In particular, when we prepare a large number of parameters for Transformer in machine translation, it is better to use these methods (and their combinations) as perturbations. We conduct more experiments to investigate whether these methods are also superior in other configurations.

4.2 High Resource

We compare each perturbation in the case where we have a large amount of training data.

Datasets We add synthetic parallel data generated from the German monolingual corpus using back-translation (Sennrich et al., 2016a) to the training data used in Section 4.1. The origin of the German monolingual corpus is NewsCrawl 2015-2018[5]. We randomly sampled 5M sentences from each NewsCrawl corpus, and thus, obtained 20M sentences in total. We back-translated the corpus

[5] data.statmt.org/news-crawl/de/

Method	Positions	2010	2011	2012	2013	2014	2015	2016	Average
w/o perturbation	-	25.63	23.62	24.54	28.39	31.50	32.96	36.47	29.02
REP(UNI)	both	26.36	24.18	25.14	28.54	**32.35**	**33.80**	37.73	29.73
REP(SIM)	both	26.04	23.79	25.01	28.43	32.06	33.28	37.40	29.43
WDROP	both	**26.65**	**24.34**	25.18	28.66	32.25	33.75	37.65	**29.78**
REP(UNI)+WDROP	both	26.45	24.07	25.09	**28.72**	32.21	33.42	37.68	29.66
REP(SIM)+WDROP	both	26.55	24.20	**25.19**	28.55	31.92	33.64	**37.96**	29.72
REP(SS)	dec	25.81	23.64	24.73	28.46	31.84	33.29	36.59	29.19
ADV	both	25.79	24.07	24.92	28.64	32.04	33.35	37.20	29.43

Table 4: BLEU scores of each method trained with a large amount of data.

with the German-English translation model, which is identical to Transformer (big) (w/o perturbation) used in Section 4.1 except for the direction of translation. Finally, we prepended a special token ⟨BT⟩ to the beginning of the source (English) side of the synthetic data following (Caswell et al., 2019). In addition, we upsampled the original bitext to adjust the ratio of the original and synthetic bitexts to 1:1.

Methods In this setting, we increase the parameter size of Transformer from the (big) setting to take advantage of large training data. Specifically, we increased the internal layer size of the FFN part from 4096 to 8192, and used 8 layers for both the encoder and decoder. The other hyper-parameters are same as in Section 4.1.

Results Table 4 shows BLEU scores of each method when we used a large amount of training data. This table indicates that all perturbations outperformed Transformer w/o perturbation in all test sets. Moreover, the fast methods REP(UNI), REP(SIM), WDROP, and their combinations achieved the same or better averaged scores than REP(SS) and ADV. Thus, these methods are not only fast but also significantly improve the performance of Transformer. In particular, since Table 3 shows that REP(UNI) and WDROP barely have any negative effect on the computational time, we consider them as superior methods.

4.3 Low Resource

Datasets We also conduct an experiment on a low resource setting. We used IWSLT 2014 German-English training set which contains 160k sentence pairs. We followed the preprocessing described in `fairseq`[6] (Ott et al., 2019). We used dev2010, 2012, and tst2010-2012 as a test set.

Methods In this setting, we reduced the parameter size of Transformer from the (base) setting. We reduced the internal layer size of the FFN part from 2048 to 1024. We used the same values for other hyper-parameters as in Section 4.1.

Method	Position	BLEU
w/o perturbation	-	35.22
REP(UNI)	both	35.53
REP(SIM)	both	35.49
WDROP	both	35.64
REP(SS)	dec	35.49
ADV	both	**36.09**
×2 training steps		
REP(UNI)	both	35.81
REP(SIM)	both	35.96
WDROP	both	36.06
REP(UNI)+WDROP	both	36.20
REP(SIM)+WDROP	both	**36.22**

Table 5: BLEU scores in the low resource setting.

Results Table 5 shows BLEU scores of each method on the low resource setting. We trained three models with different random seeds for each method, and reported the averaged scores. In this table, we also report the results of REP(UNI), REP(SIM), WDROP, and their combinations trained with twice the number of updates (below ×2 training steps). This table shows that all perturbations also improved the performance from Transformer w/o perturbation. In contrast to Tables 3 and 4, ADV achieved the top score when each model was trained with the same number of updates.

However, as reported in Section 4.1, ADV requires twice or more as long as other perturbations for training. Thus, when we train Transformer with other perturbations with twice the number of updates, the training time is almost equal. In the comparison of (almost) equal training time, WDROP achieved a comparable score to ADV. Moreover, REP(UNI)+WDROP and REP(SIM)+WDROP[7] outperformed ADV. Thus, in this low resource setting, REP(UNI)+WDROP and REP(SIM)+WDROP are slightly better than ADV in computational time.

[6] github.com/pytorch/fairseq/tree/master/examples/translation

[7] In the low resource setting, we applied only WDROP to an encoder side for REP(UNI)+WDROP and REP(SIM)+WDROP because the configuration achieved better performance than applying both perturbations to both sides.

Method	0.00	0.01	0.05	0.10
w/o perturbation	26.88	25.81	21.94	17.80
REP(UNI)	**27.57**	26.58	23.14	18.93
REP(SIM)	27.37	26.95	25.34	23.13
WDROP	27.52	26.48	22.84	18.56
REP(UNI)+WDROP	27.51	26.55	23.18	19.05
REP(SIM)+WDROP	**27.57**	**27.15**	**25.60**	**23.58**
REP(SS)	26.99	25.89	22.08	17.93
ADV	27.34	26.32	22.43	18.08

Table 6: Averaged BLEU scores on newstest2010-2016 when we inject perturbations to a source sentence with each ratio.

Method	Pos	Valid	Test	CoNLL
w/o perturbation	-	47.25	64.74	61.62
REP(UNI)	both	47.77	64.67	62.22
REP(SIM)	both	47.58	64.51	62.29
WDROP	both	48.53	65.47	62.22
REP(UNI)+WDROP	both	48.58	65.94	**62.33**
REP(SIM)+WDROP	both	**48.72**	**65.97**	62.29
REP(SS)	dec	47.84	65.18	62.30
ADV	both	48.17	65.90	62.23
Kiyono et al. (2020)	-	-	65.0	62.2

Table 7: $F_{0.5}$ scores of each method. The row of Kiyono et al. (2020) represents the reported scores of the model trained with the same configuration.

4.4 Results on Perturbed Inputs

Recent studies have used perturbations, especially adversarial perturbations, to improve the robustness of encoder-decoders (Sato et al., 2019; Cheng et al., 2019; Wang et al., 2019). In particular, Cheng et al. (2019) analyzed the robustness of models trained with their adversarial perturbations over perturbed inputs. Following them, we also investigate the robustness of our trained Transformer (big) models.

We constructed perturbed inputs by replacing words in source sentences based on pre-defined ratio. If the ratio is 0.0, we use the original source sentences. In contrast, if the ratio is 1.0, we use the completely different sentences as source sentences. We set the ratio 0.01, 0.05, and 0.10. In this process, we replaced a randomly selected word with a word sampled from vocabulary based on uniform distribution. We applied this procedure to source sentences in newstest2010-2016.

Table 6 shows averaged BLEU scores[8] of each method on perturbed newstest2010-2016. These BLEU scores are calculated against the original reference sentences. This table indicates that all perturbations improved the robustness of the Transformer (big) because their BLEU scores are better than one in the setting w/o perturbation. In comparison among perturbations, REP(SIM) (and REP(SIM)+WDROP) achieved significantly better scores than others on perturbed inputs. We emphasize that REP(SIM) surpassed ADV even though ADV is originally proposed to improve the robustness of models. This result implies that REP(SIM) is effective to construct robust models as well as to improve the performance.

[8]For more details, Tables 10, 11, and 12 in Appendix show BLEU scores on each perturbed input.

5 Experiments on GEC

Datasets Following Kiyono et al. (2020), we used a publicly available dataset from the BEA shared task (Bryant et al., 2019). This dataset contains training, validation, and test splits. We also used the CoNLL-2014 test set (CoNLL) (Ng et al., 2014) as an additional test set. We report $F_{0.5}$ score measured by the ERRANT scorer (Bryant et al., 2017; Felice et al., 2016) for the BEA dataset and M^2 scorer (Dahlmeier and Ng, 2012) for CoNLL.

Methods We used the same settings as Kiyono et al. (2020). Specifically, we trained Transformer (big) model w/o perturbation on the same parallel pseudo data provided by Kiyono et al. (2020), and then fine-tuned the model with perturbations. The hyper-parameters for perturbations are the same as those described in Section 4.1.

Results Table 7 shows the results of each method. This table reports the averaged score of five models trained with different random seeds. Moreover, we present the scores of Kiyono et al. (2020); our "w/o perturbation" model is a rerun of their work, that is, the experimental settings are identical[9].

Table 7 shows that all perturbations improved the scores except for REP(UNI) and REP(SIM) in the BEA test set (Test). Similar to the machine translation results, the simple methods WDROP, REP(UNI)+WDROP, and REP(SIM)+WDROP achieved comparable scores to ADV. Thus, these faster methods are also effective for the GEC task.

6 Related Work

Word Replacement The naive training method of neural encoder-decoders has a discrepancy between training and inference; we use the correct

[9]The scores of w/o perturbation are slightly worse than Kiyono et al. (2020). We consider that this is due to randomness in the training procedure.

tokens as inputs of the decoder in the training phase but use the token predicted at the previous time step as an input of the decoder in the inference phase. To address this discrepancy, Bengio et al. (2015) proposed the scheduled sampling that stochastically uses the token sampled from the output probability distribution of the decoder as an input instead of the correct token. Zhang et al. (2019) modified the sampling method to improve the performance. In addition, Duckworth et al. (2019) refined the algorithm to be suited to Transformer (Vaswani et al., 2017). Their method is faster than the original scheduled sampling but slower and slightly worse than more simple replacement methods such as REP(UNI) and REP(SIM) in our experiments. Xie et al. (2017) and Kobayashi (2018) used the unigram language model and neural language model respectively to sample tokens for word replacement. In this study, we ignored contexts to simplify the sampling process, and indicated that such simple methods are effective for sequence-to-sequence problems.

Word Dropout Gal and Ghahramani (2016) applied word dropout to a neural language model and it is a common technique in language modeling (Merity et al., 2018; Yang et al., 2018; Takase et al., 2018). Sennrich and Zhang (2019) reported that word dropout is also effective for low resource machine translation. However, word dropout has not been commonly used in the existing sequence-to-sequence systems. Experiments in this study show that word dropout is not only fast but also contributes to improvement of scores in various sequence-to-sequence problems.

Adversarial Perturbations Adversarial perturbations were first discussed in the field of image processing (Szegedy et al., 2014; Goodfellow et al., 2015). In the NLP field, Miyato et al. (2017) applied adversarial perturbations to an embedding space and reported its effectiveness on text classification tasks. In sequence-to-sequence problems, Wang et al. (2019) and Sato et al. (2019) applied adversarial perturbations to embedding spaces in neural encoder-decoders. Moreover, Sato et al. (2019) used virtual adversarial training (Miyato et al., 2016) in training of their neural encoder-decoders. Cheng et al. (2019) computed token-level adversarial perturbations in machine translation. In other words, they introduced the strategy of adversarial perturbations into word replacement. Their method is also effective but requires more computational time than Wang et al. (2019) and Sato et al. (2019) because it runs language models to obtain candidate tokens for perturbations.

7 Conclusion

We compared perturbations for neural encoder-decoders in view of computational time. Experimental results show that simple techniques such as word dropout (Gal and Ghahramani, 2016) and random replacement of input tokens achieved comparable scores to sophisticated perturbations: the scheduled sampling (Bengio et al., 2015) and adversarial perturbations (Sato et al., 2019), even though those simple methods are faster. In the low resource setting in machine translation, adversarial perturbations achieved high BLEU score but those simple methods also achieved comparable scores to the adversarial perturbations when we spent almost the same time for training. For the robustness of trained models, REP(SIM) is superior to others. This study indicates that simple methods are sufficiently effective, and thus, we encourage using such simple perturbations as a first step. In addition, we hope for researchers of perturbations to use the simple perturbations as baselines to make the usefulness of their proposed method clear.

Acknowledgements

We thank Motoki Sato for sharing his code with us to compare adversarial perturbations. We thank Jun Suzuki and Sosuke Kobayashi for their valuable comments. We thank anonymous reviewers for their useful suggestions. This work was supported by JSPS KAKENHI Grant Number JP18K18119 and JST ACT-X Grant Number JPMJAX200I. The first author is supported by Microsoft Research Asia (MSRA) Collaborative Research Program.

References

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (WMT)*, pages 1–61.

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 28 (NIPS), pages 1171–1179.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 793–805.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (WMT)*, pages 53–63.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4324–4333.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better Evaluation for Grammatical Error Correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 568–572.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 13063–13075.

Daniel Duckworth, Arvind Neelakantan, Ben Goodrich, Lukasz Kaiser, and Samy Bengio. 2019. Parallel scheduled sampling. *CoRR*, abs/1906.04331.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 825–835.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1019–1027.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 753–762.

Shun Kiyono, Jun Suzuki, Tomoya Mizumoto, and Kentaro Inui. 2020. Massive exploration of pseudo data for grammatical error correction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2134–2145.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 452–457.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 11432–11442.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 1–14.

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: Human Language Technologies (NAACL-HLT)*, pages 48–53.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pages 1–9.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pages 186–191.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 379–389.

Motoki Sato, Jun Suzuki, and Shun Kiyono. 2019. Effective adversarial regularization for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 204–210.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. *CoRR*, abs/1907.10597.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 211–221.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning (ICML)*, pages 5926–5936.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3645–3650.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 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 (EACL)*, pages 291–297.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

Sho Takase and Sosuke Kobayashi. 2020. All word embeddings from one embedding. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 3775–3785.

Sho Takase and Naoaki Okazaki. 2019. Positional encoding to control output sequence length. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 3999–4004.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4599–4609.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 5998–6008.

Dilin Wang, Chengyue Gong, and Qiang Liu. 2019. Improving neural language modeling via adversarial training. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6555–6565.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Levy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 9054–9065.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4334–4343.

A Experiments on Summarization

We conduct experiments on two summarization datasets: Annotated English Gigaword (Napoles et al., 2012; Rush et al., 2015) and DUC 2004 task 1 (Over et al., 2007).

A.1 Annotated English Gigaword

Datasets We used sentence-summary pairs extracted from Annotated English Gigaword (Napoles et al., 2012; Rush et al., 2015) as the summarization dataset. This dataset contains 3.8M sentence-summary pairs as the training set and 1951 pairs as the test set. We extracted 3K pairs from the original validation set, which contains 190K pairs, for our validation set.

In summarization, most recent studies used large scale corpora to pre-train their neural encoder-decoder (Dong et al., 2019; Song et al., 2019; Zhang et al., 2020; Qi et al., 2020). Thus, we also augmented the training data. We extracted the first sentence and headline of a news article in REALNEWS (Zellers et al., 2019) and News Crawl (Barrault et al., 2019) as sentence-summary pairs. In total, we used 17.1M sentence-summary pairs as our training data.

We used BPE (Sennrich et al., 2016b) to construct a vocabulary set. We set the number of BPE merge operations at 32K and shared the vocabulary between both the encoder and decoder sides.

Methods We followed the configuration in Section 4.2 because it seems suitable for a large amount of training data. We used the same perturbations and hyper-parameters as in Section 4.2.

Results Table 8 shows the ROUGE F_1 scores of each method and scores reported in recent studies (Dong et al., 2019; Song et al., 2019; Zhang et al., 2020; Qi et al., 2020) In this experiment, we cannot report the result of ADV because the loss value of ADV exploded during training. We tried several random seeds for ADV, but all models failed to converge. Since we need a huge amount of budget to search more suitable hyper-parameters for ADV in this summarization dataset, we consider that it is impractical to report the result of ADV.

Table 8 indicates that all perturbations improved the ROUGE score. In addition, REP(UNI), REP(SIM), WDROP, and their combinations outperformed the scheduled sampling. Thus, these fast methods are also superior perturbations in the summarization task. Moreover, REP(UNI) and WDROP outperformed the current top score (Qi et al., 2020) in ROUGE-1, L and ROUGE-2 respectively.

Method	Position	R-1	R-2	R-L
w/o perturbation	-	39.20	19.84	36.21
REP(UNI)	both	**39.81**	20.40	**36.93**
REP(SIM)	both	39.70	20.14	36.77
WDROP	both	39.66	**20.45**	36.59
REP(UNI)+WDROP	both	39.36	20.13	36.62
REP(SIM)+WDROP	both	39.56	20.14	36.66
REP(SS)	dec	39.20	20.04	36.27
Dong et al. (2019)	-	38.45	19.45	35.75
Song et al. (2019)	-	38.73	19.71	35.96
Zhang et al. (2020)	-	39.12	19.86	36.24
Qi et al. (2020)	-	39.51	20.42	36.69

Table 8: F_1 values of ROUGE-1, 2, and L (R-1, R-2, and R-L respectively) on the test set extracted from Annotated English Gigaword. The lower part represents the scores reported by recent studies.

A.2 DUC 2004 Task 1

Datasets We used sentence-summary pairs extracted from Annotated English Gigaword (Napoles et al., 2012; Rush et al., 2015) and News Crawl (Barrault et al., 2019) as our training dataset, which contains 10.1M pairs. Following recent studies (Takase and Okazaki, 2019; Takase and Kobayashi, 2020), we used BPE to construct a vocabulary set for the encoder side and characters as vocabulary for the decoder side. We set the number of BPE merge operations at 16K.

For the test set, we used the DUC 2004 task 1 dataset (Over et al., 2007) which contains 500 source sentences and four kinds of manually constructed reference summaries. We truncated characters over 75 bytes in each generated summary based on the official configuration.

Methods We used the Transformer (big) setting in this experiment. In addition, we introduced the output length control method proposed by Takase and Okazaki (2019). We used the same perturbations and hyper-parameters as in Section 4.1.

Results Table 8 shows recall-based ROUGE scores of each method and scores reported in recent studies (Rush et al., 2015; Suzuki and Nagata, 2017; Takase and Okazaki, 2019; Takase and Kobayashi, 2020). We also cannot report the result of ADV for the same reason as described in Appendix A.1.

This table indicates that REP(SIM), WDROP, and their combination improved the ROUGE scores. In particular, REP(SIM)+WDROP outper-

Method	Position	R-1	R-2	R-L
w/o perturbation	-	32.80	11.55	28.26
REP(UNI)	both	32.56	11.48	28.21
REP(SIM)	both	32.80	11.55	28.28
WDROP	both	**33.06**	11.45	**28.51**
REP(UNI)+WDROP	both	32.15	11.58	28.01
REP(SIM)+WDROP	both	32.80	**11.73**	28.46
REP(SS)	dec	32.83	11.41	28.14
Rush et al. (2015)	-	28.18	8.49	23.81
Suzuki and Nagata (2017)	-	32.28	10.54	27.80
Takase and Okazaki (2019)	-	32.29	11.49	28.03
Takase and Kobayashi (2020)	-	32.57	11.63	28.24

Table 9: Recall-based ROUGE-1, 2, and L (R-1, R-2, and R-L respectively) on DUC 2004 task 1 test set. The lower part represents the scores reported by recent studies.

formed the current top score in ROUGE-1, 2, and L. Moreover, WDROP achieved better ROUGE-1 and L scores than the current top score. In contrast, REP(UNI) slightly harmed the performance in this configuration. These results indicate that WDROP and REP(SIM) are also effective for summarization tasks.

Method	Positions	2010	2011	2012	2013	2014	2015	2016	Average
w/o perturbation	-	23.37	21.23	21.89	25.56	26.97	29.34	32.32	25.81
REP(UNI)	both	23.88	21.85	22.48	26.23	27.81	30.21	33.61	26.58
REP(SIM)	both	**24.51**	22.24	22.82	26.53	**28.44**	**30.43**	33.68	26.95
WDROP	both	24.01	21.90	22.48	26.24	27.60	29.71	33.44	26.48
REP(UNI)+WDROP	both	23.85	22.03	22.69	26.63	27.50	29.56	33.60	26.55
REP(SIM)+WDROP	both	24.47	**22.61**	**23.15**	**26.88**	28.24	30.14	**34.53**	**27.15**
REP(SS)	dec	23.49	21.18	21.82	25.79	27.17	29.39	32.38	25.89
ADV	both	23.94	21.70	22.46	25.99	27.71	29.28	33.13	26.32

Table 10: BLEU scores when we inject perturbations to a source sentence with 0.01.

Method	Positions	2010	2011	2012	2013	2014	2015	2016	Average
w/o perturbation	-	19.78	18.51	18.70	21.58	22.74	24.81	27.45	21.94
REP(UNI)	both	21.02	19.38	19.67	22.76	23.85	26.08	29.24	23.14
REP(SIM)	both	23.13	21.13	21.60	24.98	**26.69**	28.38	31.49	25.34
WDROP	both	20.94	19.24	19.44	22.41	23.67	25.42	28.74	22.84
REP(UNI)+WDROP	both	20.99	19.64	19.93	22.89	23.77	25.64	29.40	23.18
REP(SIM)+WDROP	both	**23.20**	**21.55**	**21.87**	**25.53**	26.50	**28.49**	**32.05**	**25.60**
REP(SS)	dec	20.06	18.58	18.90	21.92	23.01	24.59	27.51	22.08
ADV	both	20.45	18.91	19.10	22.02	23.50	24.97	28.05	22.43

Table 11: BLEU scores when we inject perturbations to a source sentence with 0.05.

Method	Positions	2010	2011	2012	2013	2014	2015	2016	Average
w/o perturbation	-	16.21	15.03	15.31	17.82	17.76	19.91	22.57	17.80
REP(UNI)	both	17.24	15.84	16.39	18.62	19.30	21.45	23.64	18.93
REP(SIM)	both	21.15	19.18	19.79	22.95	23.91	26.19	28.73	23.13
WDROP	both	16.79	15.54	16.06	18.35	18.68	20.57	23.96	18.56
REP(UNI)+WDROP	both	17.53	16.00	16.41	18.95	19.40	21.03	24.01	19.05
REP(SIM)+WDROP	both	**21.58**	**19.86**	**20.10**	**23.50**	**24.22**	**26.27**	**29.55**	**23.58**
REP(SS)	dec	16.31	15.21	15.18	18.01	18.11	20.00	22.69	17.93
ADV	both	16.47	15.24	15.50	18.01	18.07	19.84	23.44	18.08

Table 12: BLEU scores when we inject perturbations to a source sentence with 0.10.