Various Errors Improve Neural Grammatical Error Correction

Shota Koyama[1,2], Hiroya Takamura[1,2], Naoaki Okazaki[1,2]

[1]Tokyo Institute of Technology
[2]National Institute of Advanced Industrial Science and Technology
shota.koyama@nlp.c.titech.ac.jp
takamura.hiroya@aist.go.jp
okazaki@c.titech.ac.jp

Abstract

The lack of parallel data is an obstacle in data-driven grammatical error correction. Recently, researchers have addressed this problem by exploring data augmentation methods. A number of studies have focused on improving the diversity of errors in generated data. To investigate the importance of error diversity and its impact on performance improvements, we designed and integrated 188 modules to generate token-level or span-level errors. Experimental results demonstrate that the diversity of errors is crucial to performance improvements. The presented approach performs better than the baseline of round-trip translation, a purely data-driven approach. Furthermore, we report that a larger monolingual corpus does not always result in better performance.

1 Introduction

Grammatical error correction (GEC) is an important NLP application that automatically corrects mistakes in text, and can assist language learners in writing text. Typically, GEC is treated as data-driven machine translation (MT), where an erroneous sentence is 'translated' into a correct one (Brockett et al., 2006). Sequence-to-sequence models (Sutskever et al., 2014), which were originally proposed for MT, have also achieved excellent performance in GEC (Junczys-Dowmunt et al., 2018).

There are a number of publicly available datasets for GEC. The Lang-8 corpus (Mizumoto et al., 2012) is the largest among them, containing about one million sentence pairs. However, the amount of data is still insufficient for training high-quality neural machine translation (NMT) models. One promising way to address this issue is by generating additional erroneous data by adding artificial errors to grammatical sentences. Therefore, various state-of-the-art studies have developed the GEC-specific data augmentation methods in their 'pre-training and fine-tuning' paradigm (Lichtarge et al., 2019; Grundkiewicz et al., 2019; Kiyono et al., 2019).

Many studies aim at increasing the diversity in error types in augmented data, such that the model can handle a wide range of errors. Wan et al. (2020) presented a data augmentation method that injects noises to hidden-state representations based on ERRANT error types (Bryant et al., 2017). Stahlberg and Kumar (2021) used error generation models, controlling types of generated errors.

However, there are two issues that need to be discussed on the error diversity in augmented data. First, we are not certain of performance changes when error types vary. Previous studies did not analyze the contributions of diversity in error types due to its lack of explicit control in augmented data. Second, we are not certain of the performance of other important factors for improving the usefulness of artificial error data other than the variety of error types; for example, the training epochs and the size of the monolingual corpus for data augmentation.

In order to address these issues, we explore a rule-based approach for data augmentation in the GEC task. Although we are uncertain whether a purely data-driven method for data augmentation (for example, back-translation) actually generates diverse

errors, a rule-based approach allows explicit control of error types. We can evaluate the impact of error diversity by changing the combination of error-generating modules.

In this paper, we first classify errors in English text into five categories to design a wide range of modules for error generation. Next, we propose a framework for integrating the modules, in order to build artificial data with various errors. We conducted experiments to examine the impact of error diversity in the generated erroneous data. As a result, we verified that sufficient error type diversity in augmented data is important for better performance improvement. In addition to error types, we show that generating artificial data at all epochs, not only at once, and performing sufficient iterations in pre-training improve the performance even without using large monolingual corpora. Furthermore, our method achieves better performance than the baseline of round-trip translation, one of the state-of-the-art methods for data augmentation.

2 Related Work

2.1 GEC as MT

After the notable successes in machine translation, the paradigm of NMT has been widespread in GEC since Yuan and Briscoe (2016). Moreover, among a variety of sequence-to-sequence models, Transformer (Vaswani et al., 2017) has also been popular in neural GEC (Junczys-Dowmunt et al., 2018), which has advantages in translation quality and parallel computing.

2.2 Data Augmentation and Pre-Training

To make use of NMT in GEC, we need error annotated learner corpora as supervision data for NMT models. However, the size of the available supervision data in GEC is much smaller than that of NMT (Junczys-Dowmunt et al., 2018). Since preparing human-annotated GEC data is time-consuming and expensive, researchers have explored data augmentation in GEC.

Lichtarge et al. (2019) confirmed the effectiveness of pre-training an NMT model on large artificial data, and fine-tuning it on supervision data for GEC. As explained in Section 1, several researchers have explored data augmentation for GEC. Typically, data augmentation methods for GEC make artificial data by introducing errors into grammatical sentences.

Many studies have shown that simple token-level perturbations are effective for data augmentation for GEC. Grundkiewicz et al. (2019) used confusion sets obtained from a spell checker to generate artificial errors; their system was ranked first in the two tracks of the BEA-2019 shared task (Bryant et al., 2019). Choe et al. (2019) used learner's error patterns and confusion sets, based on parts-of-speech, to create preposition and inflection errors, achieving the second-best result in the same shared task.

Back-translation, where the model generates an erroneous sentence from the correct one, is an MT-based error generation method (Rei et al., 2017). Naive back-translation has a problem with the diversity of generated examples. To solve this problem, Xie et al. (2018) gave more diversity to erroneous sentences, adding noise into each hypothesis in the beam. Kiyono et al. (2019) further studied the effectiveness of back-translation in large-scale data. Stahlberg and Kumar (2021) used a Seq2Edits (Stahlberg and Kumar, 2020) model for span-level back-translation. This method can control the distribution of error types in the augmented data.

Round-trip translation (RTT), in which a grammatical sentence is first translated into a bridge language and then translated back, is also an effective MT-based error generation method (Lichtarge et al., 2019, 2020).

3 Error Categories

We categorize errors in English text into five categories to decompose various errors and investigate the importance of error diversity. This categorization is helpful to build error-generation modules, since an appropriate error generation method differs for each category. Additionally, we review previous studies on data augmentation for GEC, related to this categorization.

Function Word Errors in function words include the misuse of prepositions and pronouns; for example, *for* in "I went *for* (→to) Tokyo." Errors in function words can be generated by word-level perturbations. Izumi et al. (2003) proposed the first rule-based error generation by replacing or deleting arti-

cles. Rozovskaya and Roth (2010) replaced prepositions following the error rate and patterns of non-native writers.

Inflection Inflection errors involve inflectional forms of adjectives, nouns, and verbs; for example, *goed* in "I *goed* (→went) to Tokyo." Most inflection errors occur at word-level, although some occur at span-level (e.g., perfect tense). Brockett et al. (2006) generated inflection errors in mass nouns using regular expressions. Choe et al. (2019) produced inflection errors of nouns and verbs.

Lexical Choice Lexical choice errors occur due to the misuses of synonyms or affixes; for example, *lost* in "I *lost* (→missed) my flight." Xu et al. (2019) generated suffix errors (e.g., *arrive* →arrival) using confusion sets.

Word Order Word order errors are caused by incorrect placements of words and phrases in sentences; for example, "I *my flight missed* (→missed my flight)." Word order errors can be generated by swapping the order of adjacent words (Grundkiewicz et al., 2019).

Writing System Writing system errors include spelling, punctuation, compound, orthography, and case errors. For example, *tu* in "I went *tu* (→to) Tokyo." Grundkiewicz et al. (2019) generated confusion sets of spelling errors using a spell checker.

4 Proposed Method

In this study, we designed 188 error-generation modules and a framework for integrating them into a data augmentation method for GEC. Table 1 shows the number of modules by category. The category "Others" in the table includes modules on deleting frequent function words and punctuations, and predicting masked tokens. We first explain the details of the modules, followed by a method for integrating the modules.

4.1 Error Generation Module

The input sentence of our framework is tokenized with SpaCy v2.3[1]; each word token is tagged with a part-of-speech tag, a dependency tag, a lemma, and an IOB tag for the named entity. Error generation modules use the output of SpaCy to inject artificial errors.

Error category	# Modules
Function word	154
Inflection	5
Lexical choice	2
Word order	6
Writing system	19
Others	2
Total	188

Table 1: Breakdown of error generation modules.

Modules for function word error are designed for each function word; therefore this group occupies the majority of the modules. They delete, replace, or insert tokens based on their conditions. For example, a module on the proposition *than* deletes *than* with a probability of 0.2, and replaces *than* with *to*, *from*, *over*, *beyond* with a probability of 0.4, 0.2, 0.1, 0.1, respectively[2]. Some modules insert tokens. For example, an insertion module for articles and demonstratives inserts *a*, *an*, *the*, *this*, *that*, *these*, *those* with a probability of 0.3, 0.3, 0.3, 0.025, 0.025, 0.025, 0.025, respectively, between two adjacent words if the preceding word is `VB`, `VBD`, `VBG`, `VBN`, `VBP`, `VBZ`, or `IN` in Penn Treebank definition, and if the subsequent word is `NN`, `NNS`, `JJ`, `JJN`, or `JJS`. The module inserts articles and demonstratives at the beginning of a sentence if its first word is `NN`, `NNS`, `JJ`, `JJN`, or `JJS`.

The modules for inflection error change the inflectional form of adjectives, nouns, and verbs using `lemminflect`[3]. The other modules include the deletion of the passive auxiliary and replacement of *to* in infinitive use with *by* or *for*.

The modules for lexical choice error are for suffixes and synonyms. The suffix error module replaces a suffix with another using a manually defined suffix set. The synonym error module replaces a word with one of its synonyms using WordNet (Miller, 1995).

The modules for word order error are divided into two parts. The former part of modules move words or spans by a distance sampled from their own normal distribution. The adverb word order module and

[1] `https://spacy.io/`

[2] We set the replacing words and the probabilities manually. We may obtain further improvements by searching better parameters.

[3] `https://github.com/bjascob/LemmInflect`

the interrogative word order module move adverbs and interrogative words respectively. The module for prepositional phrase word order detects prepositional phrases and moves them. There is another special module that moves words regardless of its syntactic role. The latter part of modules swap the order of words in the detected spans. The adjective word order module detects consecutive adjectives and shuffles them. The "A of B" word order module detects the pattern of "(noun phrase) of (noun phrase)" and swaps the order of the two phrases.

The modules for writing system errors are implemented in various styles. Punctuation error modules delete, insert or replace various punctuations. Case error modules convert the first letter of a word to lowercase or uppercase. The module applies to named entities at the span-level[4]. The orthography error module randomly deletes a space between words. A variant of this module inserts a space into words based on word frequency such that, for example, "football" is more likely to be split into "foot ball" than "foo tball." The spelling error module deletes, swaps, inserts, or replaces some characters in words. The number of perturbing characters is sampled from a geometric distribution. Characters to be inserted or replaced with are sampled from a pre-defined distribution computed by a feed-forward network with a context window of five characters.

Due to space limitations, we cannot provide full explanations of the modules. Please refer to the implementation for details[5].

4.2 Integrating Error-Generation Modules

We propose a framework to integrate the presented modules. Figure 1 shows the outline of our framework. The framework consists of a stack of multiple error-generation modules, with the aim of generating diverse errors. Given a correct input sentence, it generates erroneous text artificially by applying error-generation modules one by one to the input. We manually determined the order for applying these modules, such that a subsequent module does not completely overwrite errors made by preceding modules. For example, the preposition error module precedes the spelling error modules since "on $\overset{\text{prep.}}{\rightsquigarrow}$ for $\overset{\text{spell.}}{\rightsquigarrow}$ far" is preferable to "on $\overset{\text{spell.}}{\rightsquigarrow}$ onn $\overset{\text{prep.}}{\rightsquigarrow}$ for."[6]

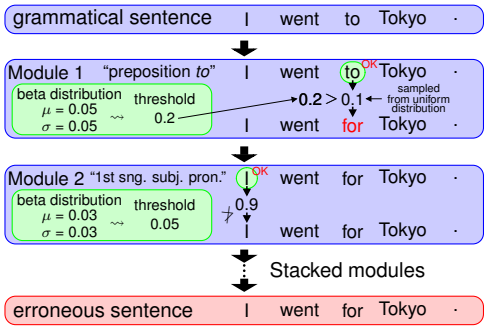

Figure 1: The outline of our error generation framework.

An error-generation module is applied to each word or span detected by the module. Each error-generation module has its own beta distribution used for sampling a threshold for each sentence[7]. For each input word, an error-generation module checks its applicability, samples the probability from the uniform distribution for each word or span, and applies to it if the probability is smaller than the threshold sampled for the sentence.

5 Experimental Settings

5.1 Training and Evaluation Dataset

We use the official datasets in Restricted Track in the BEA-2019 Shared Task (Bryant et al., 2019)[8], which consists of four training datasets with annotated errors: FCE train (Yannakoudakis et al., 2011), NUCLE (Dahlmeier et al., 2013), Lang-8 (Mizumoto et al., 2012), and W&I train (Bryant et al., 2019). We removed the identical source-target pairs in the Lang-8 corpus, and over-sampled the FCE train, NUCLE, and W&I train datasets three times.

Searching for hyperparameters on a validation set, we evaluate GEC models on: the test set of the BEA-2019 Shared Task, which is tuned for its valid set and

[4] '*long* Island' is easier to correct than '*long island*', as an erroneous example of 'Long Island'. The module addresses this issue by span-level error generation.

[5] You can see the source code for our error-generating method and all our experiments in `https://github.com/nymwa/arteraro`.

[6] Since a preposition error module refers to the original word to select an erroneous word, it is impossible to produce errors that combine prepositional and spelling errors in this example.

[7] The parameters of the beta distributions are set manually.

[8] `https://www.cl.cam.ac.uk/research/nl/bea2019st/`

ERRANT scorer (Bryant et al., 2017); the test set of the CoNLL-2014 Shared Task (Ng et al., 2014), tuned for the CoNLL-2013 Shared Task (Ng et al., 2013) dataset and M^2 scorer (Dahlmeier and Ng, 2012); the test set of FCE in the official datasets in the BEA-2019 Shared Task, tuned for its valid set and ERRANT scorer; and the test set of JFLEG (Napoles et al., 2017), tuned for its valid set and GLEU scorer (Napoles et al., 2015).

As source text for data augmentation, we use monolingual corpora in the WMT 2020 news task[9]: News crawl 2015–2020, Europarl v10, and News Commentary v16. The total number of sentences in these corpora is ca. 163.6M. When we use smaller corpora, for example, "16M sentences", we sample 16M sentences from the 163.6M sentences.

We preprocessed all the data by normalizing the punctuations and decomposable characters (e.g., à). We apply BPE-dropout (Provilkov et al., 2020) with the vocabulary size of 16,000 to all the training and evaluation data with a dropout probability of 0.1 for source sentences in training data and 0 for other data.

5.2 Model

In all experiments, we use the Transformer Big model (Vaswani et al., 2017) implemented in `fairseq` v0.10.2[10] to train GEC models. It has six Transformer blocks with 16-head self-attention layers and a hidden-vector size of 1,024. We use the GeLU activation function (Hendrycks and Gimpel, 2016) in feed-forward layers with a hidden-vector size of 4,096. We tie embeddings of the input layers of the encoder and the input and output layers of the decoder (Press and Wolf, 2017). We use a pre-norm, instead of a post-norm, for each Transformer layer for better convergence. We decode an output with a beam size of 12 and length normalization with a penalty of 0.6. We train five models for each experiment and report their average score and the score of ensemble generation.

5.3 Pre-Training and Fine-Tuning

We pre-train models using augmented data and fine-tune them using the target domain data. We call this experiment "artificial+target setting." We apply the presented error-generation framework at each epoch. This means that the supervision data for pre-training are different at each epoch. We fine-tune models for 30 epochs in artificial+target settings. The number of pre-training epochs is different for each experiment. We call the experiment without pre-training "target-only setting." We train models for 40 epochs in target-only settings.

5.4 Training Settings

We use: the cross-entropy loss with label smoothing of 0.1; AdamW optimizer (Loshchilov and Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$; a learning rate of 0.0015 for target-only settings and 0.001 for artificial+target settings; and a linear warm-up for the first 8k steps and inverted squared decay following the warm-up. The maximum length of the training sentence is 400 tokens. We use dropout for the attention and activation layers with a probability of 0.2, and other layers with that of 0.3. We apply a weight decay of 0.001 and gradient clipping of 1.0 for target-only settings and 0.3 for artificial+target settings. We set the max batch size as 4000 tokens. We accumulate gradients of 8 mini-batches for target-only settings and 128 for artificial+target settings.

6 Experimental Results

6.1 Impact of the Diversity of Error Types

To examine the importance of the diversity in error types, we investigated GEC performance by changing the error variety in the artificial data used for pre-training. Table 2 shows the results on 16M pre-training for 10 epochs and fine-tuning.

When we pre-train models using data without error generation "No error (copy)"[11], where the models are pre-trained for copying the monolingual data, the performance is worse than that of the model trained only with the target data, "Target-only." This result indicates that pre-training a model on erroneous data is necessary. The presented method "All modules" exhibits remarkable improvements over the target-only setting: 5.86, 5.46, and 2.54 points improvements (with ensembling) on BEA-19, CoNLL 14, and JFLEG, respectively.

[9] `http://www.statmt.org/wmt20/translation-task.html`

[10] `https://github.com/pytorch/fairseq`

[11] We apply gradient clipping of 0.1 for this setting to prevent divergence.

	BEA-19 test	CoNLL 14	JFLEG test
Target-only	59.09 / 63.56	54.03 / 56.79	57.33 / 58.15
Pre-training + fine-tuning, pre-trained on:			
No error (copy)	60.02 / 62.96	52.39 / 53.85	55.93 / 56.09
All modules	67.32 / 69.42	60.60 / 62.25	**60.12** / 60.69
Pre-training on errors from a specific group and others + fine-tuning			
None (others only)	64.30 / 67.60	57.88 / 59.67	58.30 / 59.00
Function word	65.79 / 68.42	58.95 / 62.14	58.58 / 58.89
Inflection	64.26 / 67.80	59.53 / 61.44	58.74 / 59.23
Lexical choice	64.91 / 67.28	58.56 / 60.87	58.68 / 59.13
Word order	64.66 / 67.76	58.60 / 60.41	58.47 / 58.76
Writing system	66.11 / 68.54	59.83 / 62.11	59.69 / 60.34
Pre-training on errors excluding a specific group + fine-tuning			
- Function word	66.57 / 69.36	59.64 / 61.62	59.79 / 60.17
- Inflection	67.03 / 69.00	**60.80** / **62.66**	60.10 / **60.82**
- Lexical choice	67.12 / **69.52**	60.47 / 61.84	**60.12** / **60.81**
- Word order	**67.60** / **69.68**	**61.28** / **63.69**	**60.12** / 60.49
- Writing system	65.79 / 68.36	60.10 / 61.62	59.28 / 59.70

Table 2: The effect of error diversity. The left and right scores represent the average score of five models and the ensemble generation score, respectively.

Using only other modules for error generation, "None (others only)" shows a considerable performance gain over the target-only setting. This result implies that mask-prediction and deletion are excellent heuristics even without linguistic features. Adding one group of modules further improves the performance. In particular, adding the modules for writing system errors is effective.

When we add four groups of modules, the results are almost identical to that of all modules, except for writing system modules. This result indicates that errors in the writing system play the most important role in the presented method. Although using all modules in error generation does not always achieve the best performance, we observe that pre-training on erroneous data of various types generally improves GEC performance.

6.2 Impact of the Corpus Size

Kiyono et al. (2019) reported that the back-translation approach for artificial error generation benefits from the larger size of text data. Figure 2 depicts the GEC performance pre-trained on artificial errors and fine-tuned on the target domain data, changing the size of the monolingual corpus from which the proposed method generates the artificial errors. The blue line presents the performance when we change the size of the monolingual corpus with the number of pre-training epochs fixed to 10. We can see that the size of the monolingual corpus matters, as reported in a previous study.

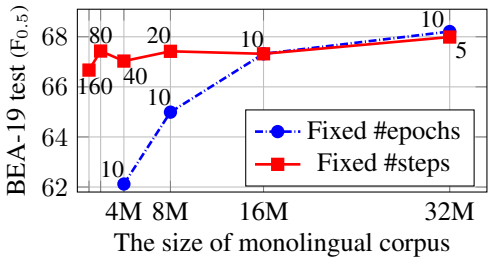

Figure 2: The effect of varying corpus size and training epochs. The results are an average of five models. The associated numbers indicate the training epochs.

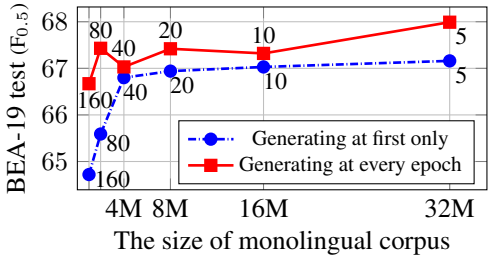

Figure 3: The effect of generating artificial data at every epoch. The results are an average of five models. The associated numbers indicate the training epochs.

However, in this experiment, we cannot distinguish whether the performance gain is caused by the increased size of monolingual corpora or by the increased total steps required to update the model parameters during pre-training. Thus, we conducted another experiment where the total number of pre-training steps was fixed, with monolingual corpus of different sizes. The red line in Figure 2 shows the result. Even when we reduced the corpus size to 1M, we did not observe a severe performance drop unlike the blue line. This result suggests that the size of monolingual corpus for artificial data is not critical, and the number of total steps for updating the model parameters during pre-training is essential. Additionally, this result implies that the proposed method can train good GEC models for low-resource languages if error-generation modules are designed appropriately.

6.3 Impact of Generating Artificial Data at Every Epoch

To investigate the effect of generating artificial error data at every epoch, we compare two settings: gen-

	BEA-19						CoNLL-13	CoNLL-14			FCE				JFLEG	
	valid			test			(valid)	(test)			valid	test			valid	test
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	$F_{0.5}$	P	R	$F_{0.5}$	$F_{0.5}$	P	R	$F_{0.5}$	GLEU	GLEU
Choe et al. (2019)	63.54	31.48	52.79	76.19	50.25	69.06	-	74.76	34.05	60.33	-	-	-	-	-	-
Grundkiewicz et al. (2019)	59.1	36.8	53.00	72.28	60.12	69.47	-	-	-	64.16	-	-	-	55.81	-	61.22
Lichtarge et al. (2020)	-	-	-	75.4	64.7	73.0	-	74.7	46.9	66.8	-	-	-	-	-	**64.9**
Wan et al. (2020)	-	-	-	72.6	61.3	70.0	-	72.3	48.8	65.9	-	65.4	53.6	**62.6**	-	-
Stahlberg and Kumar (2021)	-	-	-	77.7	65.4	**74.9**	-	75.6	49.3	**68.3**	-	-	-	-	-	64.7
Target only																
single	52.28	28.01	44.56	63.31	46.68	59.09	37.52	65.72	31.59	54.03	51.61	58.85	31.58	50.18	52.25	57.33
ensemble	58.80	27.93	48.16	69.82	46.80	63.56	37.70	71.79	30.93	56.79	54.13	63.95	31.70	53.14	52.32	58.15
ensemble + reranking	59.43	32.69	51.07	70.39	53.01	66.06	44.28	67.42	42.86	60.49	55.37	62.44	35.85	54.38	54.89	60.49
Pre-training only																
single	53.25	23.08	42.21	65.25	45.10	59.90	37.94	67.07	32.69	55.42	44.00	56.81	26.75	46.38	52.98	58.23
ensemble	54.50	23.04	42.81	66.61	45.33	60.89	38.19	67.87	32.48	55.73	44.09	57.38	26.82	46.73	53.04	58.37
ensemble + reranking	53.83	26.69	44.73	67.00	51.03	63.05	40.19	64.89	37.97	56.83	46.34	57.79	31.33	49.43	54.91	60.41
Pre-training + fine-tuning																
single	60.91	34.19	52.61	74.39	57.38	70.20	45.45	73.74	41.30	63.71	56.58	63.26	38.26	55.94	55.81	60.98
ensemble	62.65	34.12	53.67	76.46	58.17	71.93	45.88	75.76	41.26	64.91	57.38	65.19	38.45	57.23	56.13	61.34
ensemble + reranking	61.12	37.74	54.39	74.66	62.27	71.80	49.03	70.91	49.39	65.23	57.36	64.46	39.68	57.31	58.07	63.47
Pre-training + fine-tuning + fine-tuning on subset of target data																
single	58.96	41.61	54.39	72.23	63.39	70.28	-	-	-	-	57.29	64.17	40.99	57.65	-	-
ensemble	61.94	42.17	56.63	75.63	64.54	73.11	-	-	-	-	58.83	66.34	41.37	59.19	-	-
ensemble + reranking	61.73	43.13	56.83	75.03	65.28	72.85	-	-	-	-	58.63	65.38	42.10	58.87	-	-

Table 3: Comparison of the proposed method to other studies.

erating artificial data at every epoch; and generating them only once and reusing them at every epoch.

Figure 3 shows the results. We can achieve a higher performance when we generate artificial data at every epoch than we do so only once. Furthermore, the red line is less affected by the data size, whereas the blue line drops drastically when the data size is smaller than 4M. This result suggests that generating artificial data at every epoch is beneficial for diversifying erroneous sentences and improving the GEC performance regardless of the data size.

6.4 Comparison with State-of-the-Art Models

To compare our methods with state-of-the-art models, we conduct experiments using larger artificial data. We have shown that the effectiveness of the diversity in error types and generation of artificial data at every epoch. To verify the usefulness of our approach compared to the previous studies on data augmentation, we generate artificial data with 163.6M sentences 20 times, pre-train models for 20 epochs on the generated data, and fine-tune the models for 30 epochs. We accumulate gradients of 1,024 mini-batches and apply gradient clipping of 0.1 in this pre-training setting for faster training.

We explain the previous methods used for comparison. Choe et al. (2019) addressed the effectiveness of pre-training a model on artificial data, fine-tuning the model on all target data, followed by fine-tuning the model again on the subset of target training data whose domain is similar to a test set. In this setting, we fine-tune the pre-trained models on all target data for five epochs, followed by fine-tuning on the subset (BEA-19 train for BEA-19 test, and FCE train for FCE test) for 30 epochs.

Chollampatt et al. (2019) reranked the n-best candidates using the perplexity of each candidate computed by BERT (Devlin et al., 2019). We calculated the perplexity using RoBERTa large (Liu et al., 2019), following Salazar et al. (2020).

Table 3 shows the results. The "pre-training only" setting, which can be considered as an unsupervised GEC, exhibited close performances to the "target-only" setting. In other words, the proposed approach for artificial error generation is close to the fully supervised approach. The full models (the bottom group of the table) achieved comparable performance to the previous methods. The table also indicates that fine-tuning on a subset of target data, and model ensembling are quite effective in improving the performance.

6.5 Comparison to RTT

We compare our method to RTT, which was verified by Lichtarge et al. (2019) for its effectiveness. We round-trip 16M sentences using four bridge languages, German (De), Finnish (Fi), French (Fr), and Latvian (Lv), and regard pairs of original and round-tripped sentences as supervision data for pre-training. Table 4 shows the performance of the Transformer Base models that we prepared for RTT. In the RTT experiments, we pre-trained models for

BLEU	en → X	X → en
De (WMT 14)	26.6	31.8
Fi (WMT 17)	22.8	26.0
Fr (WMT 14)	37.3	35.2
Lv (WMT 17)	18.3	19.1

Table 4: BLEU scores of RTT models on WMT test data measured by `SacreBLEU` (Post, 2018).

	BEA-19 test	CoNLL 14	JFLEG test
RTT (De)	65.61 / 68.22	58.66 / 60.39	59.10 / 59.81
RTT (Fi)	65.89 / 68.25	60.09 / 62.02	59.45 / 59.65
RTT (Fr)	65.81 / 68.08	59.50 / 61.10	58.78 / 59.31
RTT (Lv)	65.80 / 67.97	58.79 / 60.51	59.20 / 59.65
RTT (all)	66.05 / 68.71	60.30 / 62.09	59.41 / 59.97
Ours	**67.32 / 69.42**	**60.60 / 62.25**	**60.12 / 60.69**

Table 5: Comparison of our method and RTT.

10 epochs using round-tripped data and fine-tuned the models as we did for the erroneous data generated by the proposed method. To diversify the RTT data, we explore the "all" setting, where a training instance is chosen from the RTT'ed ones via four languages at each epoch.

Table 5 shows the average scores of the five models. Our error-generation modules perform better than all RTT configurations including the "all" setting. We also emphasize that RTT requires much more computations than the proposed modules because RTT must use an MT system twice (fore and back translations) to obtain artificial data. In contrast, the proposed method does not require special accelerators (e.g., GPUs and TPUs) for generating data. Therefore, the presented method is not only more effective but also more efficient than RTT.

7 Conclusion

In this study, we proposed a framework to generate artificial error data for GEC, incorporating various error-generation modules. We confirmed that the diversity of errors improved the performance of GEC models pre-trained on the error data. The performance of the presented method was better than that of the RTT baselines under the same conditions. We observed that the size of the monolingual corpus was less critical, but the number of total steps in pre-training was important.

The presented error-generation modules have several hyperparameters. Optimizing these hyperparameters for a specific dataset would be an immediate future work. We further plan to combine the error-generation modules with other methods. For example, integrating error-generation modules in a decoder of back-translation and/or round-trip translation model may be a promising direction.

Acknowledgments

We thank all the anonymous reviewers for their careful reading and constructive comments. This work is supported by RWBC-OIL. For experiments, computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

References

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256.

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 (Volume 1: Long Papers)*, pages 793–805.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–227.

Shamil Chollampatt, Weiqi Wang, and Hwee Tou Ng. 2019. Cross-sentence grammatical error correction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 435–445.

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: Human Language Technologies*, pages 568–572.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 145–148.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242.

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *Transactions of the Association for Computational Linguistics*, 8:634–646.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. 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)*, pages 3291–3301.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

George A Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters*, pages 863–872.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234.

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: Shared Task*, pages 1–14.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12.

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

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 961–970.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628.

Shuyao Xu, Jiehao Zhang, Jin Chen, and Long Qin. 2019. Erroneous data generation for grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 149–158.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for au-

tomatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386.