Synthetic Data Generation for Grammatical Error Correction with Tagged Corruption Models

Felix Stahlberg and Shankar Kumar
Google Research
{fstahlberg,shankarkumar}@google.com

Abstract

Synthetic data generation is widely known to boost the accuracy of neural grammatical error correction (GEC) systems, but existing methods often lack diversity or are too simplistic to generate the broad range of grammatical errors made by human writers. In this work, we use error type tags from automatic annotation tools such as ERRANT to guide synthetic data generation. We compare several models that can produce an ungrammatical sentence given a clean sentence and an error type tag. We use these models to build a new, large synthetic pre-training data set with error tag frequency distributions matching a given development set. Our synthetic data set yields large and consistent gains, improving the state-of-the-art on the BEA-19 and CoNLL-14 test sets. We also show that our approach is particularly effective in adapting a GEC system, trained on mixed native and non-native English, to a native English test set, even surpassing real training data consisting of high-quality sentence pairs.

1 Introduction

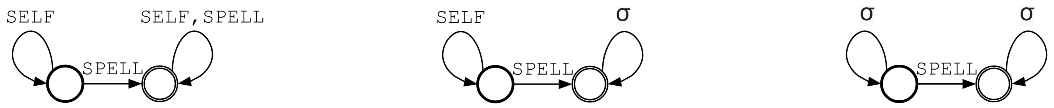
Grammatical error correction (GEC) systems aim to automatically correct grammatical and other types of writing errors in text. It is common to view this problem as a sequence-to-sequence task (i.e. ungrammatical sentence → grammatical sentence) and borrow models that were originally developed for neural machine translation (NMT) (Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018; Ge et al., 2018b). Back-translation (Sennrich et al., 2016) is a synthetic data generation technique for NMT that employs a translation system trained in the reverse direction to synthesize source sentences from sentences in the target language, and is still one of the most effective strategies to use monolingual data in NMT training. Similarly, synthetic training data generation for GEC has also been widely studied in the literature (Brockett et al., 2006; Foster and Andersen, 2009; Rozovskaya and Roth, 2010; Felice et al., 2014; Rei et al., 2017; Kasewa et al., 2018; Xie et al., 2018; Ge et al., 2018a,b; Kiyono et al., 2019; Lichtarge et al., 2019; Stahlberg and Byrne, 2019; Zhao et al., 2019; Xu et al., 2019; Grundkiewicz et al., 2019; Choe et al., 2019; Takahashi et al., 2020). This work is inspired by previous efforts to use ideas from back-translation for GEC (Kasewa et al., 2018; Xie et al., 2018; Kiyono et al., 2019). In contrast to prior work, we use error type tags such as `SPELL` (spelling error) or `SVA` (subject-verb agreement error) to control the output of our corruption models and generate more realistic as well as diverse grammatical errors. Our tagged corruption models are trained to output the corrupted sentence given a clean sentence and an error tag, e.g.:

> "`NOUN:INFL` There were a lot of sheep."
> → "There were a lot of sheeps."

The tags mitigate the tendency of untagged corruption models to produce simplistic corruptions since many error type tags require more complex rewrites. In general, there is a one-to-many mapping from a clean sentence to a noisy sentence. Using a regular corruption model, many of these synthetic errors tend to be simplistic,[1] but adding tag information allows the model to generate specific patterns of errors that can be found in actual GEC corpora. The benefit of covering a wide range of error types when generating pseudo data for GEC has also been demonstrated by Takahashi et al. (2020); Wan et al. (2020). Moreover, the tag distribution in the synthetic data can be made to match the distribution of a specific target domain. We use this distribution matching technique to adapt a GEC system to better correct errors by native speakers.

[1] Example outputs from untagged and tagged corruption models can be found in Appendix B.

Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications, pages 37–47
April 20, 2021 ©2021 Association for Computational Linguistics



(a) NoSigma (b) PostSigma (c) PrePostSigma

Figure 1: Constraining FSTs for the `SPELL` tag. The σ-self loops can match any tag. `SELF` is used by Seq2Edits to mark source spans that are not modified.

Tag	Log-prob.	Corruption model output
ADJ	-3.06	There were a lot of many sheep.
ADJ:FORM	-2.49	There were a more better of sheep.
ADV	-2.63	There were a lot of sheep there.
CONJ	-1.39	And there were a lot of sheep.
CONTR	-0.90	There're a lot of sheep.
DET	-1.06	There were lot of sheep.
K	-1.45	There were a lot of.
MORPH	-0.67	There were a lot of sheeps.
NOUN	-3.31	There were a lot of seep.
NOUN:INFL	-0.61	There were a lot of sheeps.
NOUN:NUM	-1.33	There were a lots of sheep.
NOUN:POSS	-0.92	There were a lot of sheep's.
ORTH	-0.79	There were alot of sheep.
OTHER	-2.60	There were many sheep.
PART	-1.22	There were a lot off sheep.
PREP	-1.08	There were a lot sheep.
PRON	-1.55	It was a lot of sheep.
PUNCT	-1.00	There were a lot of sheep
SPELL	-2.79	There were a lot of sheeps.
VERB	-2.58	There had a lot of sheep.
VERB:FORM	-2.34	There being a lot of sheep.
VERB:INFL	-1.09	There were a lot of sheeps.
VERB:SVA	-0.44	There was a lot of sheep.
VERB:TENSE	-0.57	There are a lot of sheep.
WO	-1.87	There were a lot sheep of.

Table 1: Outputs of a tagged Seq2Edits corruption model for the example input sentence "There were a lot of sheep.". The ERRANT error type tags are described in Bryant et al. (2017).

As an alternative to adding the tag directly to the input sequence, we add inference-time constraints to the recently proposed Seq2Edits model (Stahlberg and Kumar, 2020) to force the generation of a particular tag. We implement such constraints using Finite State Transducers (FSTs). We then use these corruption models to generate synthetic training data that follows a desired tag distribution, for example the tag distribution on the development set. Using our new synthetic pre-training sets[2] we report state-of-the-art results on two popular GEC test sets (BEA-test: 74.9 $F_{0.5}$, CoNLL-14: 68.3 $F_{0.5}$). In our experiments on GEC for native English, a model fine-tuned on synthetic data that follows a native English error-tag distribution can even surpass a model fine-tuned on high-quality, real (i.e. non-synthetic) data.

2 Tagged Corruption Models

At the core of our approach is a model that generates an ungrammatical sentence from a clean sentence given an error tag $t \in \mathcal{T}$ that describes the desired type of error. $\mathcal{T}$ is the set of 25 error type tags supported by the automatic annotation toolkit ERRANT (Felice et al., 2016; Bryant et al., 2017).

A straightforward way to train such a tagged corruption model is to annotate a parallel corpus with ERRANT, prepend the ERRANT tag to the clean sentence, and train a model such as a standard Transformer (Vaswani et al., 2017) to generate the ungrammatical sentence.[3] This idea is similar to the multi-lingual NMT system of Johnson et al. (2017) that adds the target language ID tag to the source sentence.

Alternatively, the recently proposed Seq2Edits[4] (Stahlberg and Kumar, 2020) model is able to directly predict error tags along with the edits, and does not need to be provided error tags in the input sequence. Instead, during beam search we constrain the tag output tape of Seq2Edits with an FST that forces the generation of a certain tag. Fig. 1 illustrates three types of constraint FSTs with the example tag, `SPELL`. All FSTs require at least one occurrence of the `SPELL` tag. NoSigma (Fig. 1a) is the most restrictive constraint as it only allows `SPELL` and `SELF` (used by Seq2Edits for unmodified source spans). PostSigma (Fig. 1b) allows other tags after `SPELL`, but constrains the hypothesis to start with either `SELF` or `SPELL` to prevent beam search from committing to a corruption that

[2] The data set will be made publicly available.

[3] If a sentence pair has multiple tags we duplicate it in the training set for each unique tag. This potentially enables the corruption model to learn co-occurrences of error categories since multiple errors may be labelled with a single tag.

[4] A short description of the Seq2Edits model is provided in Appendix A for convenience.

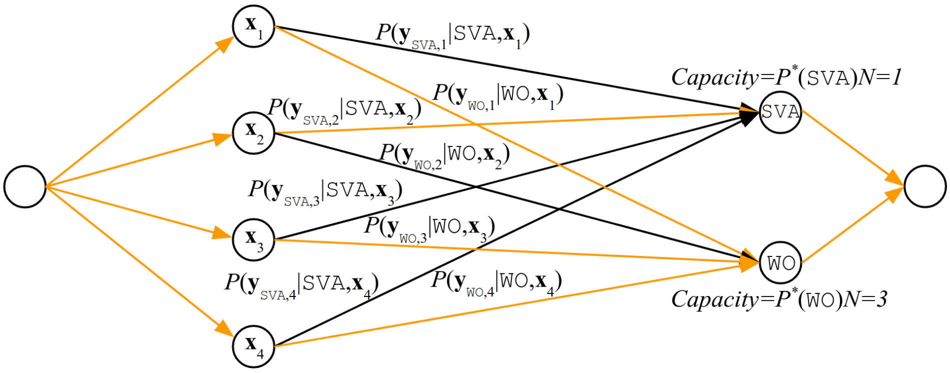


Figure 2: The minimum-cost flow graph of the *Offline-Optimal* method for a training set of $N = 4$ sentences, two error type tags (SVA: subject-verb agreement, and WO: word order) and the desired tag distribution $P^*(\texttt{SVA}) = 0.25$ and $P^*(\texttt{WO}) = 0.75$. Each sentence is represented by a node on the left side with a flow capacity of one. Each tag is represented by a node on the right side with a capacity equal to the absolute number of sentences for each tag $P^*(t)N$. Arcs connecting sentences and tags are weighted by the corruption model score $P(\mathbf{y}_{t,n}|t, \mathbf{x}_n)$. A possible flow through the graph is highlighted in orange, this assigns the WO tag to the sentences $\mathbf{x}_1$, $\mathbf{x}_3$, $\mathbf{x}_4$ and the SVA tag to the sentence $\mathbf{x}_2$.

is incompatible with SPELL.[5] PrePostSigma (Fig. 1c) allows other tags both before and after SPELL.

Table 1 lists example outputs of a tagged Seq2Edits corruption model for all 25 ERRANT tags and demonstrates the model's capability to generate a broad variety of realistic errors.

3 Synthetic Data Generation with Tagged Corruption Models

For a grammatical input sentence $\mathbf{x}_n$ ($n \in [1, N]$ where N is the training set size), we denote the corrupted sentence according to the tag $t \in \mathcal{T}$ as $\mathbf{y}_{t,n}$. Our goal is to assign a single tag t^*_n to each training sentence such that the overall distribution follows a certain desired tag distribution $P^*(t)$:

$$\forall t \in \mathcal{T} : P^*(t) \approx \frac{|\{t^*_n = t|n \in [1, N]\}|}{N} \qquad (1)$$

We compare three different methods: *Offline-Optimal*, *Offline-Probabilistic*, and *Online*.

Offline-Optimal The *Offline-Optimal* method frames this task as a constrained optimization problem:

$$\max_{\mathbf{t}^*} \sum_{n=1}^{N} \log P(\mathbf{y}_{t^*_n,n}|t^*_n, \mathbf{x}_n) \qquad (2)$$

under the constraint that the observed distribution of tags matches the desired distribution, i.e. Eq. 1 is satisfied. Fig. 2 illustrates that this is an instance of a well-studied problem called maximum weighted bipartite matching (Schrijver, 2003) and can be solved efficiently with a standard minimum-cost flow solver.

Offline-Probabilistic The intuition behind the *Offline-Probabilistic* method is to first draw a tag according to the desired tag distribution $P^*(t)$ and then sample sentences which are most likely to contain this tag, i.e. draw N sentences from the distribution $P((\mathbf{x}, \mathbf{y})|t)$.

$$\begin{aligned} P((\mathbf{x}, \mathbf{y})|t) &= \frac{P(\mathbf{x}, \mathbf{y})P(t|\mathbf{x}, \mathbf{y})}{\sum_{n=1}^{N} P(\mathbf{x}_n, \mathbf{y}_n)P(t|\mathbf{x}_n, \mathbf{y}_n)} \\ &= \frac{P(t|\mathbf{x}, \mathbf{y})}{\sum_{n=1}^{N} P(t|\mathbf{x}_n, \mathbf{y}_n)}, \end{aligned}$$

where we assume each sentence-pair has the same probability $P(\mathbf{x}, \mathbf{y}) = \frac{1}{N}$.

$$\begin{aligned} P(t|(\mathbf{x}, \mathbf{y})) &= \frac{P(t, \mathbf{x}, \mathbf{y})}{P(\mathbf{x}, \mathbf{y})} \\ &= \frac{P(\mathbf{x})P(t|\mathbf{x})P(\mathbf{y}|t, \mathbf{x})}{P(\mathbf{x}, \mathbf{y})} \\ &\approx \frac{\frac{1}{N}\frac{1}{|\mathcal{T}|}P(\mathbf{y}|t, \mathbf{x})}{\frac{1}{N}} \\ &= \frac{1}{|\mathcal{T}|}P(\mathbf{y}|t, \mathbf{x}), \end{aligned}$$

where we assume that a) each sample has equal probability i.e. $P(\mathbf{x}) \approx P(\mathbf{x}, \mathbf{y}) = \frac{1}{N}$, b) each tag is equally likely given the source sentence, $P(t|\mathbf{x}) = \frac{1}{|\mathcal{T}|}$, where $|\mathcal{T}|$ is the size of the tag

[5] An example of this garden-path problem would be a subject-verb-agreement (SVA) constraint, but all active hypotheses in the beam already contain an adjective error and the correct subject and verb (e.g.: "SVA He owns a large bike with tiny wheels" → "He owns a wide bike with...").

Data set	Synthetic	Number of sentences	Used for
WikiEdits (Lichtarge et al., 2019)	No	170M	Pre-training
RoundTripGerman (Lichtarge et al., 2019)	Yes	176M	Pre-training
Lang-8 (Mizumoto et al., 2011)	No	1.9M	Stage 1 fine-tuning
FCE-train (Yannakoudakis et al., 2011)	No	26K	Stage 2 fine-tuning
BEA-train (Bryant et al., 2019)	No	34K	Stage 2 fine-tuning
This work: C4$_{200M}$	Yes	200M	Pre-training

Table 2: Training data sets used in this work.

vocabulary. $P(\mathbf{y}|t, \mathbf{x})$ is the probability assigned by the corruption model to the target sequence $\mathbf{y}$ given the source $\mathbf{x}$ and tag t.

Unlike the *Offline-Optimal* approach, this method does not guarantee that each sentence from the original training set will be included in the sample. However, this may not matter in practice when drawing from a large pool of examples.

Online A major limitation that prevents the *Offline-Optimal* and *Offline-Probabilistic* methods from scaling up efficiently is that we need to run the corruption model for every combination of tag and source sentence ($\Omega(N|\mathcal{T}|)$ runtime).[6] The *Online* method avoids this computational complexity by drawing the tag t_n^* for each example from the desired tag distribution $P^*(\cdot)$, and then generating the target $\mathbf{y}_n$ given the source $\mathbf{x}_n$ and tag t_n^*.

$$\forall n \in [1, N] : t_n^* \sim P^*. \qquad (3)$$

Thus, it does not rely on the corruption model probabilities. The *Online* method assigns tags on-the-fly to each sentence independently and hence runs in $\Theta(N)$.

4 Results

For comparability to related work, we report span-based ERRANT $F_{0.5}$-scores on the development and test sets (BEA-dev and BEA-test) of the BEA-2019 shared task (Bryant et al., 2019). We use the M2 scorer (Dahlmeier and Ng, 2012) to compute $F_{0.5}$-scores on the CoNLL-13 (Ng et al., 2013) and CoNLL-14 (Ng et al., 2013) sets, and the GLEU metric (Napoles et al., 2015) on JFLEG-dev and JFLEG-test (Napoles et al., 2017).

4.1 Training Setup

All our grammar *correction* models are standard Seq2Seq (*not* Seq2Edits) Transformers (Vaswani et al., 2017) trained with Adafactor (Shazeer and Stern, 2018) using the Tensor2Tensor (Vaswani et al., 2018) TensorFlow (Abadi et al., 2015) library. Our *corruption* models are either standard Transformers or Seq2Edits models (Stahlberg and Kumar, 2020).[7] We use a Tensor2Tensor joint 32K subword vocabulary and the 'Big' hyper-parameter set for all our models. For our tagged corruption models we extend the subword vocabulary by the 25 ERRANT error tags.

We use both existing and new data sets to train our models (Table 2). WikiEdits and RoundTripGerman are large but noisy pre-training sets described by Lichtarge et al. (2019). In this work we introduce a new synthetic pre-training corpus – C4$_{200M}$ – that we generated by applying our corruption methods to 200M sentences sampled randomly from the Colossal Clean Crawled Corpus (Raffel et al., 2020, C4).[8] Our final *correction* models are trained using the two stage fine-tuning recipe of Lichtarge et al. (2020): after pre-training we first fine-tune on Lang-8 (Mizumoto et al., 2011) and then on BEA+FCE which is the combination of the FCE corpus (Yannakoudakis et al., 2011) and the training split of the Cambridge English Write & Improve corpus used in the BEA-2019 shared task (Bryant et al., 2019). Our *corruption* models are trained using a similar setup but do not use C4$_{200M}$ in pre-training. In our ablation experiments, however, we modify specific stages of this training pipeline to gain more insight into our methods.

4.2 Synthetic vs. Real Parallel Data

In initial experiments (Tables 3 to 5) we explore how well our synthetic data generation methods can replace real parallel data. The corruption models used in this section are fine-tuned on Lang-8 but not on BEA+FCE. The seed correction model is pre-trained on WikiEdits and RoundTripGerman and fine-tuned on Lang-8. We discard the source sentences in BEA+FCE, replace them with synthetic

[6] We ignore the runtime of beam search when describing the asymptotic time complexity for simplification.

[7] The focus of our work was to examine techniques for synthetic data correction while keeping the *correction* model fixed. Hence, we do not use Seq2Edits models for *correction*.

[8] We filtered C4 with language ID and removed sentences longer than 250 words before selecting the 200M sentences.

Corruption model	Constraint	Tagged input?	Offline-Optimal			Offline-Probabilistic			Online		
			P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
a Full sequence	-	✓	54.5	16.7	37.5	51.0	20.7	39.4	53.5	18.5	38.8
b Seq2Edits	NoSigma		57.2	21.8	43.2	57.1	25.6	45.8	57.5	26.4	46.6
c Seq2Edits	PostSigma		56.5	21.6	42.7	56.6	25.7	45.6	59.4	27.2	48.0
d Seq2Edits	PrePostSigma		56.3	22.4	43.3	55.7	25.5	45.0	53.0	30.6	46.2
e Seq2Edits	-	✓	55.3	26.7	45.5	55.6	25.9	45.2	54.4	29.0	46.3

Table 3: *Tagged* synthetic data generation where tags are chosen according to the BEA-dev tag distribution. The GEC seed model (Table 4a) is fine-tuned on BEA+FCE with synthetic source sentences and evaluated on the BEA-dev set. Rows **a** and **e** prepend the desired tag to the input sequence while rows **b**-**d** use FST constraints.

GEC data (2. fine-tuning)		BEA-dev		
Source	Target	P	R	$F_{0.5}$
a N/A (Seed model)		57.0	12.8	33.7
b Real data	BEA+FCE	56.5	35.2	50.4
c Synthetic (Full seq.)	BEA+FCE	57.6	20.6	42.4
d Synthetic (Seq2Edits)	BEA+FCE	53.4	20.6	40.5

Table 4: *Untagged* synthetic data generation. The seed GEC model (row **a**) is fine-tuned on BEA+FCE target sentences that are either paired with the real source sentences (row **b**) or with back-translated source sentences using either a full sequence Transformer (row **c**) or a Seq2Edits (row **d**) corruption model.

System	Test set ($F_{0.5}$)			
	A2	B2	C2	N2
Baselines				
a Seed model	37.6	34.1	31.4	22.2
b FT on real data	**50.3**	**51.5**	**44.1**	42.1
Synthetic data using target tag distributions $P^*(t)$				
c CEFR-A (A1)	47.4	46.2	39.0	39.0
d CEFR-B (B1)	47.1	46.0	40.9	38.0
e CEFR-C (C1)	47.1	46.2	37.1	39.1
f Native (N1)	47.8	49.2	42.8	**42.9**

Table 5: Adapting GEC to non-native or native English. In rows **c**-**f** the GEC seed model (row **a**) is fine-tuned on BEA+FCE with source sentences synthesized by a tagged Seq2Edits corruption model by following proficiency-dependent tag distributions (A1, B1, C1, or N1). FT denotes fine-tuning.

corruptions of the target sentences, and fine-tune the seed correction model on this synthetic data, i.e. all models in Tables 3 to 5 are trained by fine-tuning the same seed model (Table 4a and 5a) on the same set of target sentences but different sets of source sentences.

Data generation without tags Fine-tuning the seed model on the real parallel data improves the $F_{0.5}$-score on BEA-dev by 16.7 points (33.7 → 50.4 in rows **a** and **b** of Table 4). Our goal is to close the gap relative to the $F_{0.5}$ of 50.4 using synthetic source sentences. The corruption models in rows **c** and **d** of Table 4 do not use any error tags, which is similar to previous attempts to apply back-translation to GEC (Kasewa et al., 2018).

Data generation with tags Table 3 reports results from the tag-based corruption methods introduced in this work. Seq2Edits (rows **b**-**e**) is more amenable to tag-based corruption than a standard full sequence Transformer model (row **a**) because tag prediction is a component of the Seq2Edits model. Interestingly, the *Offline-Optimal* method tends to perform worse than *Offline-Probabilistic* and *Online* in the constrained Seq2Edits experiments (rows **b**-**e**). We hypothesize that *Offline-Optimal* might generate duller and more systematic errors because the corruption model score is used to pair tags with sentences. Increasing the diversity of synthetic errors by selecting non-optimal tag-sentence pairs ultimately improves the usefulness of the synthetic data.[9]

Comparing Table 4 with Table 3 we observe that controlling the tag distribution of the synthetic data from a Seq2Edits model outperforms traditional back-translation without tags. Our best model (*Online* column in Table 3c) achieves an $F_{0.5}$-score of 48.0 which is much better than our best system without tags (42.4 $F_{0.5}$ in Table 4c) and remarkably close to the oracle score of 50.4 $F_{0.5}$ (Table 4b) obtained using a model trained on real parallel data.

For all experiments in the remainder of this paper we used the unconstrained tagged Seq2Edits corruption models (Table 3e) because it yields reasonable gains across all methods (*Offline-Optimal*, *Offline-Probabilistic*, and *Online*) and is easiest and most practical to run on a large scale.[10] Furthermore, we will only use the *Online* method to avoid the computational overhead of *Offline-Optimal* and *Offline-Probabilistic*.

[9] The same intuition motivates our experiments in Sec. 4.3 that replace beam search with sampling.

[10] We noticed that constrained decoding (Table 3b-d) often fails in large-scale experiments if the selected tag and source sentence are incompatible.

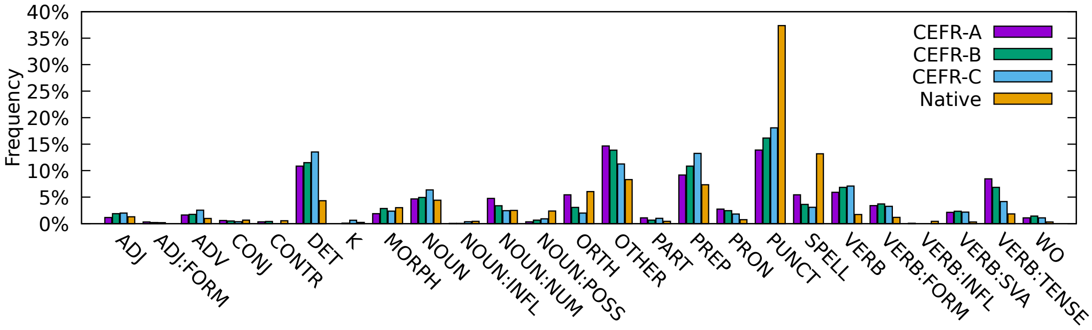


Figure 3: ERRANT error tag distributions on BEA-dev for non-native (CEFR levels A, B, C) and native English.

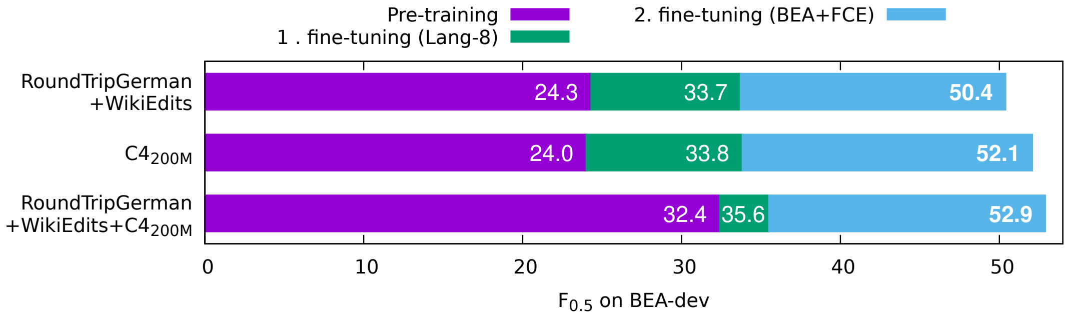


Figure 4: Using C4$_{200M}$ (tagged Seq2Edits corruption model, BEA-dev tag distribution) in pre-training. The 2-stage fine-tuning setup is described in Sec. 4.1.

Adapting GEC to English proficiency levels A potential use case for tagged corruption models is to adapt a GEC model to the proficiency level of the user by changing the target tag distribution $P^*(t)$ of the synthetic fine-tuning set. Each sentence in the BEA-dev development set is annotated with English proficiency labels (CEFR-levels A, B, C, and 'N' for native English). We split BEA-dev using these labels, and then split each set again into two parts (development/test), resulting in eight disjoint subsets A1, A2, B1, B2, C1, C2, N1, N2 with about 500 sentences each. We use A1, B1, C1, and N1 to estimate proficiency-dependent target tag distributions. As before we fine-tune the seed model on the BEA-train target sentences with synthetic source sentences that follow one of these tag distributions, but evaluate the fine-tuned models on the A2, B2, C2, and N2 splits. Table 5 shows that the tag distributions from A1, B1, and C1 yield similar performance (rows **c-e** in Table 5) across most test sets. This suggests that our method is not effective at discriminating between the different CEFR-levels of non-native English. However, using the tag distribution from native speakers (N1 in Table 5f) does yield substantial gains on the native English test set (42.9 $F_{0.5}$ on N2), even surpassing the real parallel data (42.1 $F_{0.5}$ Table 5b). This demonstrates the potential of tag-based corruption for improving GEC of native English.

Fig. 3 shows that the error tag distribution for native English differs significantly from the non-native distributions. Native speakers (orange bar) tend to make more punctuation (PUNCT), and spelling (SPELL) mistakes whereas the determiner errors (DET) are more common in non-native text.

4.3 The C4$_{200M}$ Synthetic Data Set

We showed in the previous section that using an unconstrained tagged Seq2Edits corruption model that follows the BEA-dev tag distribution works well in a controlled setup (corrupting ~60K clean target sentences from BEA+FCE). We now apply the same corruption model to a much larger, clean data set (C4$_{200M}$) consisting of 200M sentences and use the resulting synthetic data set as an additional pre-training set for our GEC models. Fig. 4 reports performance from three different GEC models with different pre-training sets, each using the 2-stage fine-tuning pipeline described in Sec. 4.1. The RoundTripGerman+WikiEdits model resembles the baseline of Lichtarge et al. (2020). Using C4$_{200M}$ instead of RoundTripGerman+WikiEdits

Tag distribution $P^*(t)$	Decoding	BEA-dev P	BEA-dev R	BEA-dev $F_{0.5}$	CoNLL-13 P	CoNLL-13 R	CoNLL-13 $F_{0.5}$	JFLEG-dev GLEU
a None (no tags)	Beam search	58.1	36.5	51.9	58.5	29.0	48.6	57.2
b BEA-dev	Beam search	58.0	39.3	52.9	58.5	33.0	50.7	57.9
c CoNLL-13	Beam search	58.8	39.6	53.6	58.9	33.3	51.0	57.9
d JFLEG-dev	Beam search	58.3	39.1	53.1	58.9	31.6	50.2	57.6
e Uniform	Beam search	59.1	39.6	53.8	58.3	33.4	50.7	57.7
f None (no tags)	Sampling	57.5	36.0	51.4	56.9	29.4	47.9	57.1
g BEA-dev	Sampling	59.5	**41.3**	**54.7**	**59.2**	**34.7**	**51.9**	**58.5**
h CoNLL-13	Sampling	58.6	40.7	53.9	58.5	33.3	50.8	58.1
i JFLEG-dev	Sampling	59.0	39.7	53.8	58.2	34.0	51.0	58.4
j Uniform	Sampling	**59.6**	40.6	54.5	58.3	34.3	51.1	58.3

Table 6: Using different target tag distributions to corrupt $C4_{200M}$ with a tagged Seq2Edits corruption model, either using beam search or sampling. We report the best of five training runs after two-stage fine-tuning according to the performance on the development set.

GEC data (pre-training) Synthetic source	Target	BEA-dev P	BEA-dev R	BEA-dev $F_{0.5}$
a Untagged corruption	WikiEdits	40.8	4.0	14.3
b RoundTripGerman	WikiEdits	33.6	11.6	24.3
c Tagged corruption	WikiEdits	39.7	20.2	33.3

Table 7: Data generation on WikiEdits for pre-training. Models are pre-trained on a mix of the original WikiEdits data set and a synthetic data set that consists of WikiEdits target sentences corrupted with either (1) an untagged Seq2Edits corruption model (row **a**), (2) round-trip translation via German (row **b**, or row 1 in Fig. 4), or (3) a tagged Seq2Edits corruption model (row **c**) following the BEA-dev tag distribution.

improves the final $F_{0.5}$-score to 52.1. Combining all three pre-training sets leads to a large jump in $F_{0.5}$ to 32.4 after pre-training. The gains are reduced after fine-tuning, but our best model still uses all three pre-training sets (52.9 $F_{0.5}$ after the second fine-tuning stage). The gains in Fig. 4 from using $C4_{200M}$ can be attributed to a) the tagged corruption method, or b) the use of C4 rather than Wikipedia which covers a broader range of text types. In the ablation experiment in Table 7, rather than using $C4_{200M}$, we corrupted the WikiRevision target sentences with various corruption methods. Tagged corruption (row **c**) outperforms both untagged corruption (row **a**) and round-trip translation (row **b**) when the target sentences are kept constant.

A crucial practical question is whether our approach is sensitive to the particular target tag distribution $P^*(t)$, and if the synthetic $C4_{200M}$ training data can help generalization to other development sets. Rows **b**-**e** in Table 6 show the performance after fine-tuning for four different tag distributions: BEA-dev, CoNLL-13, JFLEG-dev, and Uniform. Each row reports the performance of a model pre-trained using RoundTripGerman+WikiEdits and $C4_{200M}$ corrupted using the desired tag distribution followed by the 2-stage fine-tuning, i.e. row **b** corresponds to row 3 in Fig. 4. All tagged corruption models improve upon the untagged models (rows **a** and **f**).[11] In contrast to our adaptation experiments in Table 5, the variations between different tag distributions are small. This indicates that even though choosing the *correct* tag distribution is crucial for adapting GEC to native English, at the pre-training stage the ability of tagged corruption models to generate diverse errors is more important than matching a particular distribution.

Previous work on back-translation has found that it can be beneficial to use sampling instead of beam search for synthetic data generation (Edunov et al., 2018; Kiyono et al., 2019). We confirm these findings for our tagged corruption models: Sampling (Table 6g-j) outperforms beam search (Table 6b-e) for all tag distributions except CoNLL-13. Using sampling and the BEA-dev tag distribution (Table 6g) yields good performance across all development sets. The BEA-dev tag distribution reflects a wide range of grammatical errors across various proficiency levels compared to other corpora such as CoNLL-14 (mostly beginner) or FCE (School) (Bryant et al., 2019). Table 8 situates this single model and an ensemble of five analogously trained models in the context of related work. For our final models in Table 8 we follow Lichtarge et al. (2019, 2020); Stahlberg and Kumar (2020) and multiply the model score of the identity mapping with a factor (tuned on the development set) to balance precision and recall.[12] Our single model outperforms other single models on CoNLL-14 and JFLEG-

[11] For more insight into the difference between untagged and tagged corruptions see Appendix B.

[12] This factor is around 1.0 for BEA-dev and CoNLL-13 (i.e. no impact) but it helps to re-balance precision and recall on JFLEG (around 2.0).

System	BEA-test			CoNLL-14			JFLEG-test
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	GLEU
Single systems							
Kiyono et al. (2019)	65.5	59.4	64.2	67.9	44.1	61.3	59.7
Omelianchuk et al. (2020)	**79.2**	53.9	**72.4**	**77.5**	40.1	65.3	-
Lichtarge et al. (2020)	67.6	62.5	66.5	69.4	43.9	62.1	63.8
Kaneko et al. (2020)	67.1	60.1	65.6	69.2	45.6	62.6	61.3
Wan et al. (2020)	66.9	60.6	65.5	69.5	47.3	63.5	-
This work	72.1	**64.4**	70.4	72.8	**49.5**	**66.6**	**64.7**
Ensembles							
Grundkiewicz et al. (2019)	72.3	60.1	69.5	-	-	64.2	61.2
Kiyono et al. (2019)	74.7	56.7	70.2	72.4	46.1	65.0	61.4
Omelianchuk et al. (2020)	**79.4**	57.2	73.7	**78.2**	41.5	66.5	-
Lichtarge et al. (2020)	75.4	64.7	73.0	74.7	46.9	66.8	**64.9**
Kaneko et al. (2020)	72.3	61.4	69.8	72.6	46.4	65.2	62.0
Wan et al. (2020)	72.6	61.3	70.0	72.3	48.8	65.9	-
This work	77.7	**65.4**	**74.9**	75.6	**49.3**	**68.3**	64.7

Table 8: Comparison of our final system with related work.

test. Our ensemble establishes new state-of-the-art scores on BEA-test (74.9 $F_{0.5}$) and CoNLL-14 (68.3 $F_{0.5}$). We would like to emphasize that these gains are achieved without modifying the GEC model architecture – our GEC models are vanilla Transformers that were pre-trained using our new synthetic $C4_{200M}$ data set. We will make our data set publicly available to make it easy for other researchers to benefit from our work. Appendix C contains example outputs from our system trained with $C4_{200M}$ that demonstrate improved fluency and better handling of long-range reorderings.

5 Related Work

The body of literature on synthetic data generation for GEC is large. Various heuristics have been proposed to inject synthetic noise into grammatical sentences such as random word- or character-level insertion, substitution, deletion, or shuffling operations (Lichtarge et al., 2019; Zhao et al., 2019; Xu et al., 2019), using spell checkers (Grundkiewicz et al., 2019), or randomly applying word edits extracted from the training data (Choe et al., 2019). Kasewa et al. (2018); Stahlberg and Byrne (2019) applied back-translation (Sennrich et al., 2016) to GEC and reported substantial gains. Similar to MT (Edunov et al., 2018), back-translation for GEC can be further improved by adding noise to the decoding process (Xie et al., 2018) or by using sampling instead of beam search (Kiyono et al., 2019). Fluency-boost learning (Ge et al., 2018a,b) can also be used to generate additional sentence pairs during training. Lichtarge et al. (2019) proposed to generate noisy counterparts of grammatical English sentences by translating them to another language (e.g. German) and back ("round-trip translation"), a technique we also use in this work. The use of tags for back-translation in MT has been explored by Caswell et al. (2019). Our tagged corruption models are inspired by Wan et al. (2020) who generated synthetic sentences from latent representations that are perturbed using explicit error type tags. Our approach of adding the tags to the input sequence is simpler as it requires no modifications to the model architecture or training procedure.

6 Conclusion

We have introduced a synthetic data generation method for grammatical error correction that is able to produce a wide range of realistic grammatical errors. Our method is based on grammar corruption models that corrupt a clean sentence given an error type tag. Conditioning on the error type tag enables us to control synthetic data generation much more precisely than alternative methods such as round-trip translations or tag-independent back-translation. We explored different ways of using these tagged corruption models to generate synthetic data that follows a certain error tag distribution. We found that fine-tuning a model on synthetic data that follows a native English error tag distribution can even outperform fine-tuning on genuine parallel data from a mixture of proficiency levels. Along with this paper we publish a new 200M sentence data set for GEC – $C4_{200M}$. Using $C4_{200M}$ in pre-training of vanilla Transformer GEC models yields state-of-the-art performance on two standard GEC test sets (BEA-test and CoNLL-14). We expect this corpus to further stimulate the development of new data-driven approaches in GEC.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

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, Sydney, Australia. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

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, Vancouver, Canada. Association for Computational Linguistics.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

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, Montréal, Canada. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan. The COLING 2016 Organizing Committee.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Jennifer Foster and Øistein E. Andersen. 2009. Generrate: Generating errors for use in grammatical error detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, EdAppsNLP '09, pages 82–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065, Melbourne, Australia. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

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, Florence, Italy. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. Association for Computational Linguistics.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium. Association for Computational Linguistics.

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, Hong Kong, China. Association for Computational Linguistics.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

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, Beijing, China. Association for Computational Linguistics.

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, Valencia, Spain. Association for Computational Linguistics.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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, Copenhagen, Denmark. Association for Computational Linguistics.

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, Cambridge, MA. Association for Computational Linguistics.

A. Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Felix Stahlberg and Bill Byrne. 2019. The CUED's grammatical error correction systems for BEA-2019. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 168–175, Florence, Italy. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Yujin Takahashi, Satoru Katsumata, and Mamoru Komachi. 2020. Grammatical error correction using pseudo learner corpus considering learner's error tendency. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 27–32, Online. Association for Computational Linguistics.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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, New Orleans, Louisiana. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. 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 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.