Mitigating Exposure Bias in Grammatical Error Correction with Data Augmentation and Reweighting

Hannan Cao* **Wenmian Yang†** **Hwee Tou Ng***
* Department of Computer Science, National University of Singapore
† School of Computer Science and Engineering, Nanyang Technological University
caoh@u.nus.edu, wenmian.yang@ntu.edu.sg, nght@comp.nus.edu.sg

Abstract

The most popular approach in grammatical error correction (GEC) is based on sequence-to-sequence (seq2seq) models. Similar to other autoregressive generation tasks, seq2seq GEC also faces the exposure bias problem, i.e., the context tokens are drawn from different distributions during training and testing, caused by the teacher forcing mechanism. In this paper, we propose a novel data manipulation approach to overcome this problem, which includes a data augmentation method during training to mimic the decoder input at inference time, and a data reweighting method to automatically balance the importance of each kind of augmented samples. Experimental results on benchmark GEC datasets show that our method achieves significant improvements compared to prior approaches.[1]

1 Introduction

Grammatical error correction (GEC) is the task of correcting errors in a source sentence and generating a well-written and grammatically correct target sentence. Good results have been achieved by state-of-the-art GEC systems (Rothe et al., 2021; Stahlberg and Kumar, 2021) based on the sequence-to-sequence (seq2seq) transformer (Vaswani et al., 2017) architecture.

Generally, the seq2seq models are optimized to predict the next token given all previous context tokens at each time step. During training, the ground truth tokens are used as the context, i.e., the so-called teacher forcing mechanism. However, during inference, the context is the previous tokens predicted by the model. As a result, the contexts at training and inference time are drawn from different distributions. This discrepancy is called *exposure bias* (Ranzato et al., 2016), which causes the model to make inferences under conditions that it has not met during training and causes errors to accumulate during inference.

Source	If that also word for you, pleas close the ticker.
Target	If that also works for you, please close the ticket.
Decoder input	
Actual	If that also work for you, pleas close the ticker.
Noise injection	**It** that also **works** for you, pleas **cloze** the ticker.
Random sampling	If that also **works** for you, **please** close the ticker.

Table 1: Example decoder inputs produced by different methods. Each method's deviations from the actual decoder input are shown in **bold**.

Currently, the exposure bias problem has attracted much attention in autoregressive text generation. For example, in neural machine translation (NMT), some work (Ranzato et al., 2016; Mihaylova and Martins, 2019; Zhang et al., 2019) addresses this problem by utilizing sentence-level metrics, randomly injecting noise into the training samples as the augmented decoder input, or randomly replacing tokens in the training samples with predicted tokens as the augmented decoder input. Although GEC is a text generation task like NMT, it emphasizes more on proposing correct edits. Therefore, feeding artificially augmented sentences into decoder input during training may produce a different distribution compared with the actual decoder input at inference time.[2] As a result, those generated errors may never happen, making the augmented sentences less effective in the GEC task. As illustrated in Table 1, the augmented decoder input using noise injection or random sampling differs from the actual decoder input.

[1]The source code of this paper is publicly available at https://github.com/nusnlp/gec_eb

[2]We empirically demonstrated the difference in Table 5 in Section 4.4

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2123–2135

To address the exposure bias problem more effectively in GEC, we propose a data augmentation approach, which reduces the difference in distribution of the samples between training and inference in an intuitive way. Specifically, we first collect the beam search output generated by the GEC model as the augmented sentences, which mimic the actual decoder input at inference time. Then, during training, the ground-truth and augmented sentences are both fed as input to the decoder, thus reducing the difference in distribution of the samples between training and inference.

Moreover, previous work (Zhang et al., 2019; Bengio et al., 2015) shows that adjusting the noise density during training can better address the exposure bias problem. Since different beam search candidates contain different noise densities (i.e., the error density in GEC), and affect the model performance to varying extent, each candidate should be treated unequally. Therefore, we further propose a data-reweighting method, which automatically learns sampling probability for different augmented sentences dynamically. Specifically, our reweighting approach learns a data scorer through reinforcement learning, which learns the sampling probability for each augmented sentence.

The contributions of our paper are as follows:

- We empirically demonstrated that the exposure bias problem exists in the top-performing seq2seq GEC models, and addressing them helps to improve performance.
- We propose a novel data manipulation approach in GEC, including a data augmentation approach that mitigates the distribution gap between training and inference, and a data reweighting approach that automatically learns the sampling probability for each kind of training samples. No previous work uses data reweighting to tackle exposure bias.
- To the best of our knowledge, our data manipulation approach is the first work that addresses the exposure bias problem regardless of the evaluation metrics in seq2seq GEC.
- Experimental results show that our data augmentation approach significantly reduces the exposure bias problem and improves the performance of seq2seq GEC models.

2 Related Work

This section briefly introduces recent related research in the literature.

2.1 Exposure Bias

Exposure bias has been mostly studied in NMT. Bengio et al. (2015) replace the ground truth word by sampling randomly from the previous predicted words during training. Zhang et al. (2019) further inject noise into the training examples at both word level and sentence level, and select sentence level candidates based on the BLEU score (Papineni et al., 2002). Instead of injecting random noise to the ground truth sentences during training, our data augmentation method treats model-generated sentences as the augmented decoder input to mitigate the difference in distribution of samples between training and inference.

Ranzato et al. (2016) utilize Mixed Incremental Cross-Entropy Reinforce to optimize the model using metrics used at test time (e.g., BLEU) directly. Many similar works (Shao et al., 2018; Saunders et al., 2020) have also been proposed in NMT. For GEC, Sakaguchi et al. (2017) address the exposure bias problem by using reinforcement learning to directly optimize the GEC model towards the GLEU score (Napoles et al., 2015). Compared to Sakaguchi et al. (2017), we do not optimize the GEC model using reinforcement learning, and we address the exposure bias problem effectively where GLEU score is not used.

2.2 Data Reweighting

Many methods have been developed for reweighting or selecting a subset of data suitable for training a model (Franceschi et al., 2018; Chen et al., 2017; Wu et al., 2018; Shu et al., 2019; Wang et al., 2020; Lichtarge et al., 2020). Specifically, Lichtarge et al. (2020) propose an offline reweighting method for GEC, which reweights each training sentence in the previous stage of training based on the perplexity difference between the current and previous checkpoint. Wang et al. (2020) use a reweighting method to learn the sampling distribution for different languages in multilingual machine translation. In contrast, our method is an online reweighting method which dynamically adjusts the error density of the augmented sentences according to different model states.

3 Method

In this section, we first briefly introduce a baseline GEC system in Section 3.1. Then, we introduce our data manipulation method in Section 3.2.

3.1 A Baseline GEC system

Let x be the ungrammatical source sentence and y be the corrected grammatical target sentence. For a seq2seq GEC model parameterized by θ, the goal is to minimize the negative log-likelihood (NLL) for a set of M sentence pairs $\{\langle x^{(i)}, y^{(i)}\rangle\}_{i=1}^{M}$, as follows:

$$l(x, d, y; \theta) = -\sum_{t=1}^{L_y}\sum_{k=1}^{|V|} \mathbb{1}\{y_t = k\} \cdot \log P(y_t = k | d_{<t}, x; \theta) \quad (1)$$

$$L_{NLL}(\theta) = \frac{1}{M}\sum_{i=1}^{M} l(x^{(i)}, y^{(i)}, y^{(i)}; \theta) \quad (2)$$

where x is the source sentence, d is the decoder input of the seq2seq model, y is the target sentence, L_y is the length of the target sentence, $|V|$ is the vocabulary size of the target language, y_t is the t-th target token, $\mathbb{1}\{\cdot\}$ is the indicator function, and $P(\cdot|\cdot)$ is the conditional probability with model θ.

Given trained parameters $\hat{\theta}$, the hypothesis sentence $\hat{y}$ is generated using beam search to select the candidate with the highest probability, as follows:

$$\hat{y} = \arg\max_{y}\{P(y|x; \hat{\theta})\} \quad (3)$$

3.2 Data Manipulation

We propose a data manipulation method that mitigates the difference in contexts during training and inference to improve performance. In particular, our method consists of two parts: A data augmentation method that generates augmented sentences based on beam search to mimic the decoder input distribution during inference, and a data reweighting method that assigns sampling probability for augmented sentences. The overview of our approach is shown in Figure 1.

We will first introduce the data augmentation method in Section 3.2.1 and then introduce the data reweighting method in Section 3.2.2.

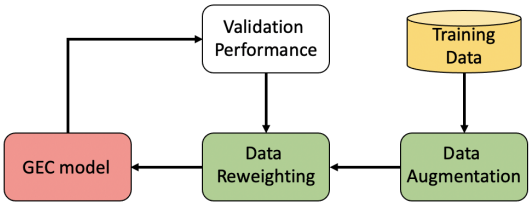

Figure 1: Overview of the proposed data manipulation method to mitigate exposure bias in GEC.

3.2.1 Data Augmentation

To reduce the distribution gap of decoder input during training and inference, we propose to add augmented sentences into the decoder input during training. These augmented sentences are generated by beam search using the model parameters from the checkpoint of the previous stage[3], which mimic the decoder input at inference time.

More specifically, we generate augmented sentences by feeding the source sentence x to the model parameterized by θ, and choose the top n output sentences $\{y_j\}_{j=1}^{n}$ with the highest probabilities generated by beam search.

As the decoder input must have the same length as the target during training, we first align the augmented sentences $\{y_j\}_{j=1}^{n}$ with the corresponding target sentence y according to linguistically-enhanced Damerau-Levenshtein alignment (Felice et al., 2016) by the ERRANT toolkit.[4] Then, we remove the excess tokens and add paddings for the missing tokens. The padded augmented sentences are denoted as $\{\bar{y}_j\}_{j=1}^{n}$. We collect $(x, \{\bar{y}_j\}_{j=1}^{n}, y)$ to form an augmented dataset D_{aug}.

Intuitively, for a well-trained GEC model, candidates with higher probability to be generated usually have lower error densities[5]. Therefore, different candidates affect the model performance unequally. To better capture the effect of different augmented sentences, we assign a separate weight for each augmented sentence, and the loss function is formulated as follows:

$$L_{EB}(\theta) = l(x, y, y; \theta) + \sum_{j=1}^{n} \alpha_{x,j} \cdot l(x, \bar{y}_j, y; \theta) \quad (4)$$

where $\alpha_{x,j}$ represents the weight for the jth augmented sentence for source sentence x. We set

[3]Following prior work (Stahlberg and Kumar, 2021), we train the GEC model in three stages, and load the checkpoint of the second stage

[4]https://github.com/chrisjbryant/errant

[5]We verified this in Table 5 in Section 4.4

$\sum_{j=1}^{n} \alpha_{x,j} = 1$, so that all the augmented sentences are treated equally as the original sentence.

3.2.2 Data Reweighting

The ideal way to train an optimal GEC model using Eq. 4 is to determine the optimal value for each $\alpha_{x,j}$. However, this can be computationally costly. To efficiently weight each augmented sentence, we use the averaged weight γ_j to approximate each individual weight $\alpha_{x,j}$, where $\gamma_j \sim E_{\alpha_j}$ and E_{α_j} represents the expectation of $\alpha_{x,j}$, calculated by

$$E_{\alpha_j} = \mathbb{E}_j[\alpha_{x,j}, x \in D_{train}]$$

Therefore, we reformulate Eq. 4 as:

$$L_{EB}(\theta) = l(x, y, y; \theta) + \sum_{j=1}^{n} \gamma_j \cdot l(x, \bar{y}_j, y; \theta) \tag{5}$$

Previous work (Zhang et al., 2019; Bengio et al., 2015) has demonstrated that varying noise density (error density in GEC) according to the training state of the model can effectively address the exposure bias problem. Therefore, we use the sampling probability β_j to approximate γ_j and propose to dynamically adjust β_j according to the model θ. Specifically, we design a learnable data scorer $P(j; \psi)$, parameterized by ψ to modify the training objective which we use to learn θ, so as to maximize the validation performance. Here $j \in \{1, ..., n\}$ indicates the index of the augmented sentence (index 1 indicates the sentence generated with the highest probability in beam search, and so on). More specifically, in the training process, $P(j; \psi)$ is used as the probability to sample an augmented sentence at index j such that the validation loss is minimized.

The final objective is written as follows:

$$\begin{aligned} \psi^* &= \operatorname*{argmin}_{\psi} L(\theta^*(\psi), D_{dev}) \text{ where} \\ \theta^*(\psi) &= \operatorname*{argmin}_{\theta} \sum_{(x,d,y)} L_{EB}(\theta, \psi) \end{aligned} \tag{6}$$

and $L_{EB}(\theta)$ in Eq. 5 is reformulated as:

$$\begin{aligned} L_{EB}(\theta, \psi) = l(x, y, y; \theta) + \\ \sum_{j=1}^{n} \mathbb{1}\{j = s_i\} \cdot l(x, \bar{y}_j, y; \theta) \end{aligned} \tag{7}$$

where $s_i \in \{1, ..., n\}$ represents the sampled index based on the learned probability $P(j; \psi)$ for each index j [6].

[6] We use torch.multinomial function from the PyTorch package to sample one candidate at a time.

During training, we optimize θ and ψ iteratively. We first optimize the data scorer ψ with fixed θ. To update the data scorer, we use reinforcement learning with a reward function that approximates the effect of the training data on the model's development set performance.

Specifically, our *environment* is the model state θ and model input (x, d, y). Our *agent* is the data scorer network ψ. We formulate our *reward* as:

$$\begin{aligned} &R(j; \theta_t) \\ &= \cos(\nabla_\theta L_{dev}(D_{dev}; \theta_t) \cdot \nabla_\theta L_{train}(j; \theta_t)) \end{aligned} \tag{8}$$

where $L_{dev}(D_{dev}; \theta_t)$ denotes the validation loss calculated by Eq. 2, $L_{train}(j; \theta_t)$ represents the training loss for different kinds of augmented sentences, and $\cos(\cdot)$ denotes the cosine similarity of the two vectors.

Algorithm 1: Data reweighting method

Input: D_{train}, D_{dev}, D_{aug}
Output: Optimal parameter θ^*

Initialize θ, ψ
while *not converge* **do**
  Sample f batches of training data $(X, D, Y) \sim (D_{train}, D_{aug})$
  Sample validation data $(X_{dev}, Y_{dev}) \sim D_{dev}$
  $g_{dev} \leftarrow \sum_{(x_{dev}, y_{dev}) \in (X_{dev}, Y_{dev})} \nabla_\theta l(x_{dev}, y_{dev}, y_{dev}; \theta) / |(X_{dev}, Y_{dev})|$
  ▷ Update ψ
  for (x, d, y) *in* (X, D, Y) **do**
    ▷ Estimate different j effect
    for j *in* $\{1, ..., n\}$ **do**
      $g_\psi \leftarrow \nabla_\theta L_{train}(j; \theta)$
      $R(k; \theta) \leftarrow \cos(g_{dev}, g_\psi)$
    end
    ▷ Optimize ψ
    $d_\psi \leftarrow \sum_{j \in \{1,...,n\}} R(j; \theta) \cdot \nabla_\psi \log P(j; \psi)$
    $\psi \leftarrow \text{GradientUpdate}(\psi, d_\psi)$
  end
  ▷ Update θ
  for (x, d, y) *in* (X, D, Y) **do**
    $g_\theta \leftarrow \nabla_\theta L_{EB}(\theta, \psi)$
    $\theta \leftarrow \text{GradientUpdate}(\theta, g_\theta)$
  end
end

To capture the combined effect of augmented sentence and the original training sentence to the model's performance, we calculate $L_{train}(j;\theta_t)$ as follows:

$$L_{train}(j;\theta_t) = l(x,y,y;\theta) + l(x,\bar{y}_j,y;\theta_t) \quad (9)$$

Intuitively, this reward makes the data scorer upweight the training data that have a similar gradient direction as the development data (Wang et al., 2020).

According to the REINFORCE algorithm (Williams, 1992), the update rules for the data scorer become:

$$\psi_t \leftarrow \psi_{t-1} + R(j;\theta_t) \cdot \nabla_\psi \log P(j;\psi) \quad (10)$$

After obtaining ψ, we update θ by:

$$\theta_t \leftarrow \theta_{t-1} - \nabla_\theta \sum_{(x,d,y)} L_{EB}(\theta,\psi) \quad (11)$$

The training algorithm is shown in Algorithm 1. Note that we do not calculate the validation gradient in every training step, since it is too computationally expensive for the GEC task. Instead, we calculate the validation gradient for every f steps. Specifically, we first sample validation data and f batches of training data from the validation and the training set, respectively. Then, we obtain validation gradient g_{dev} based on the current model weights θ. For each batch of training data, we further calculate the reward for the data scorer using Eq. 8 and update the data scorer using the REINFORCE algorithm. After updating the data scorer, we apply the output from ψ to Eq. 7 and use this sampled loss to update the model θ with the same f batches of training data.

In this paper, we use a single-layer embedding network followed by a softmax layer as the data scorer networks ψ, and a transformer-based framework as our model θ. More details are given in Section 4.2.

4 Experiments

In this section, we report the effectiveness of our data manipulation approach.

4.1 Data and Model Configuration

Following (Kaneko et al., 2020; Stahlberg and Kumar, 2021), we build our GEC model based on the transformer-big architecture (Vaswani et al., 2017). To compare with state-of-the-art approaches in GEC, which pre-train with synthetic data, we follow (Stahlberg and Kumar, 2021) to generate $C4_{200M}$ which contains 200 million synthetic parallel sentences. We use $C4_{200M}$ to pre-train our GEC model, and use the combination of NUCLE (Dahlmeier et al., 2013), FCE (Yannakoudakis et al., 2011), and CLang-8 (Rothe et al., 2021) to train our GEC model, using an encoder-decoder shared vocabulary of 10K byte pair encoding tokens.[7]

We evaluate the effectiveness of our data manipulation method on two datasets: the CoNLL-2014 test set (Ng et al., 2014) and the BEA-2019 test set (Bryant et al., 2019). The W&I training set is used for fine-tuning. The development dataset is the CoNLL-2013 dataset (Ng et al., 2013) (W&I dev) when evaluating on the CoNLL-2014 test set (BEA-2019 test set).

4.2 Experimental Setup

We build our GEC model using the fairseq toolkit (Ott et al., 2019). All experiments are carried out on one NVIDIA A100 GPU. We use the MaxMatch scorer (Dahlmeier and Ng, 2012) to evaluate performance on the CoNLL-2014 test set and the ERRANT scorer (Bryant et al., 2019) to evaluate performance on the BEA-2019 test set. We report the average scores of three runs, and use a one-tailed sign test with bootstrap resampling to carry out statistical significance tests. We generate the augmented samples with a beam size of 5, and choose all five predictions for each training sentence as augmented samples for both CoNLL-2014 test set and BEA-2019 test set.

Baselines. We evaluate the performance of our method by comparing against five baseline methods. We apply our method and all the baseline methods to the checkpoint before fine-tuning.[8] **CE:** Fine-tuning the model using cross entropy loss. **SS:** Fine-tuning the model using the scheduled sampling method (Bengio et al., 2015). **RN-Word:** Fine-tuning the model by randomly injecting noise at word level (Zhang et al., 2019). **RN-Sent:** Fine-tuning the model by our modified version of sentence-level noise injection (Zhang et al., 2019) to better adapt to the GEC task. Specifically, we choose a noisy sentence with the smallest edit-

[7] More training details are given in Appendix A.7.

[8] The model checkpoint has been pre-trained first on $C4_{200M}$ and then further trained on the combination of NUCLE, FCE, and CLang8.

System	CoNLL-14	BEA-19
Single System		
(Kiyono et al., 2019)†	61.3	64.2
(Lichtarge et al., 2020)†	62.1	66.5
(Wan et al., 2020)†	63.5	65.5
(Stahlberg and Kumar, 2021)†	66.6	70.4
(Rothe et al., 2021)*	68.9	75.9
(Sun et al., 2021)#	66.4	72.9
DM†	66.8	71.5
Ensemble		
(Kiyono et al., 2019)	65.0	70.2
(Wan et al., 2020)	65.9	70.0
(Lichtarge et al., 2020)	66.8	73.0
(Stahlberg and Kumar, 2021)	68.3	74.9
DM	68.5	74.8

Table 2: $F_{0.5}$ scores of state-of-the-art seq2seq GEC systems on the CoNLL-2014 and BEA-2019 test sets. #: Variant of BART-large model; *: T5-xxl model; †: transformer-big model.

distance, instead of the highest BLEU score, as the sentence-level oracle. We show the difference in Appendix A.3. **Ad-Bridge:** Fine-tuning the model by the AdapBridge method (Xu et al., 2021) which adaptively injects noise on word-level with word-level matching score. **RE-DP:** Weighting each augmented sentence using delta-perplexity during fine-tuning (Lichtarge et al., 2020).

4.3 Experimental Results

Approach	CoNLL-2014			BEA-2019		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
CE	70.3	51.0	65.4	70.8	67.7	70.2
SS	70.7	51.0	65.6	71.6	67.2	70.7
RN-Word	71.1	51.0	65.9	71.5	67.6	70.7
RN-Sent	71.8	50.8	66.3	71.7	67.3	70.8
Ad-Bridge	71.4	50.5	66.0	71.0	67.6	70.4
DA	73.6	47.4	66.3*	72.6	66.0	71.1
+RE-DP	73.7	47.5	66.4	72.8	65.8	71.3
DM	74.0	48.1	**66.8***†	73.1	65.5	**71.5**

Table 3: Experimental results (in %) on the CoNLL-2014 and BEA-2019 test sets. DA represents fine-tuning the model using only the data augmentation approach mentioned in Section 3.2.1 where each γ_j in Eq. 5 is set to $1/n$. DM represents fine-tuning the model using our proposed data manipulation method (including both data augmentation and data reweighting). Statistically significant improvements ($p < 0.01$) over CE and DA are marked as * and † respectively. Since the BEA-2019 test set is a blind test set, we are unable to carry out statistical significance tests on it.

The performance of our data manipulation method is shown in Table 3.[9] All methods which aim to mitigate the exposure bias problem (i.e., SS, RN-Word, RN-Sent, Ad-Bridge, DA, and DM) outperform CE on both CoNLL-2014 and BEA-2019 test sets. This shows the importance of addressing the exposure bias problem in GEC.

Furthermore, our DM method outperforms all baseline methods on both test sets. Specifically, on the CoNLL-2014 test set, DM achieves an $F_{0.5}$ score of 66.8%, which is 0.5% higher than RN-Sent. On the BEA-2019 test set, DM achieves an $F_{0.5}$ score of 71.5%, which improves by 0.7% compared to RN-Sent. DM performs better since it is able to better approximate the decoder input distribution at inference time and adjust the amount of errors automatically based on the model θ.

We also compare DM with state-of-the-art seq2seq GEC systems in Table 2. Specifically, we show the performance of a single DM model and an ensemble of eight analogously trained DM models. Among all the transformer-big single systems, our method achieves the best $F_{0.5}$ scores on both CoNLL-2014 and BEA-2019 test sets, surpassing (Stahlberg and Kumar, 2021) by 0.2% and 1.1%, respectively. Although Rothe et al. (2021) achieve $F_{0.5}$ scores of 68.9% and 75.9% on the CoNLL-2014 and BEA-2019 test set respectively, they adapt the T5-xxl model with 11 billion parameters, whereas our transformer-big model only uses 220 million parameters (2% of the size of T5-xxl model). The corresponding $F_{0.5}$ scores of (Rothe et al., 2021) for their T5-base model with

[9]The results on the CWEB and JFLEG test sets are shown in Appendix A.8.

Example 1	
Source	~~Many~~ Much of this surveillance ~~are~~ is being implemented in the government sectors ~~,~~ or military areas to enhance their security.
CE	Many of this surveillance are being implemented in the government sectors ~~,~~ or military areas to enhance their security.
RN-Sent	Many of this surveillance ~~are~~ is being implemented in the government sectors ~~,~~ or military areas to enhance their security.
DM	Many of this surveillance ~~are~~ is being implemented in the government sectors ~~,~~ or military areas to enhance their security.
Example 2	
Source	Surveillance technology will help to prevent ~~the family~~ families ~~to~~ from ~~loss~~ losing their ~~member~~ members, especially the elderly and the children ~~which~~ who need to be ~~pay~~ paid more attention to.
CE	Surveillance technology will help to prevent ~~the~~ family to loss their member, especially the elderly and the children ~~which~~ who need to be pay more attention to.
RN-Sent	Surveillance technology will help to prevent ~~the~~ family to loss their ~~member~~ members, especially the elderly and the children ~~which~~ who need to be pay more attention to.
DM	Surveillance technology will help to prevent ~~the~~ family ~~to~~ from ~~loss~~ losing their ~~member~~ members, especially the elderly and the children ~~which~~ who need to be pay more attention to.

Table 4: Examples of the error accumulation problem, with the predictions taken from CE, RN-Sent, and DM models. Red texts show the correct edits.

220 million parameters are 65.1% and 69.4% for the CoNLL-2014 and BEA-2019 test set respectively.

Our DM ensemble achieves an $F_{0.5}$ score of 68.5% on the CoNLL-2014 test set and reaches an $F_{0.5}$ score of 74.8% on the BEA-2019 test set, which is close to the best performance achieved by (Stahlberg and Kumar, 2021). Note that Stahlberg and Kumar (2021) have used more than 540 million synthetic sentence pairs to pre-train their transformer big model, while we only use 200 million synthetic sentence pairs. Note that the current best $F_{0.5}$ scores achieved by an ensemble approach (Qorib et al., 2022) on the CoNLL-2014 and BEA-2019 test sets are 69.51% and 79.90% respectively. However, we have not included them in Table 2 as Qorib et al. (2022) combine sequence-to-sequence models with sequence-tagging models.

4.4 Effect of Reweighting

We analyze the sampling distribution for each augmented sentence in Figure 2, and show the error density (in terms of the number of edits required to correct to grammatical sentence) for each augmented sentence in Table 5. Specifically, the error density is measured by the average number of edits required to transform an augmented sentence into the target sentence.

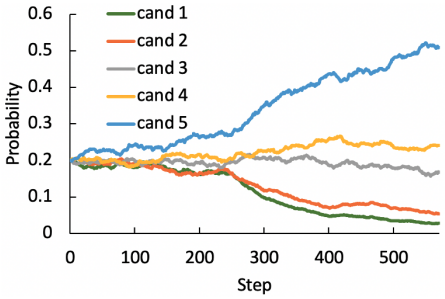

Figure 2: The sampling probability of augmented sentences during training.

As shown in Table 5 and Figure 2, the sampling probability for candidates with higher error density keeps increasing during training. This shows that as training progresses, candidates with higher error density play a more important role in training the model, which supports the hypotheses from (Zhang et al., 2019; Bengio et al., 2015).

Compared to RE-DP[10] which uses a static error density to train the model, our DM method achieves better performance by dynamically adjusting the error density according to the model θ. This further confirms that a varying error density is more

[10]We show the detailed settings of RE-DP in Appendix A.2.

Sentence	err%	e/s
WI train	67.1	3.45
BS-1	54.8	2.64
BS-2	90.7	2.18
BS-3	95.6	2.24
BS-4	96.9	2.30
BS-5	98.5	2.39
RN-Sent	-	7.82

Table 5: Statistics of the W&I training set and different augmented sentences. BS-x denotes the sentence which is the beam search candidate at position x. e/s is the number of edits per sentence calculated on erroneous sentences. err% is the percentage of erroneous sentences in the entire set. The err% for RN-Sent varies according to the different training epochs, with the details given in Appendix A.1.

effective in addressing the exposure bias problem.

4.5 Effect on Exposure Bias

In this section, we first present representative examples to illustrate the error accumulation problem in GEC. Then we conduct comprehensive studies to show the effectiveness of our method in addressing the exposure bias problem.

4.5.1 Effect on Error Accumulation

In Example 1 from Table 4, the CE model fails to correct “Many” to “Much” (since “surveillance” is an uncountable noun, “Much” should be used instead of “Many”). Failing to correct this error causes the CE model to also fail to correct “are” to “is”, which illustrates the error accumulation problem. Although RN-Sent and DM model also fail to correct “Many” to ‘Much”, they successfully correct “are” to “is”. This shows that both RN-Sent and DM are better able to address the error accumulation problem caused by exposure bias.

In the second example, the output generated by CE, RN-Sent, and DM all fail to correct “family” to “families”. For the output of CE, this further causes the model to fail to correct “to loss” to “from losing” and “member” to “members”. For RN-Sent, failing to correct “family” to “families” causes the model to fail to correct “to loss” to “from losing”, but it successfully corrects “member” to “members”. For the DM model, it is able to recover from failing to correct “family” to “families” and successfully corrects both “to loss” to “from losing” and “member” to “members”. Therefore, the error accumulation problem is better addressed by our DM approach.

Source	Her impressive physique ~~asa~~ as well as her ~~extraorinbary~~ extraordinary faculties as a rumba dancer will not be forgotten.
DA	Her impressive physique asa well as her extraorinbary faculties as rumba dancer will not be forgotten.
DM	Her impressive physique asa well as her ~~extraorinbary~~ extraordinary faculties as a rumba dancer will not be forgotten.

Table 6: Examples of the error accumulation problem, with the predictions taken from the DA and DM model. Red texts show the correct edits.

In Table 6, we show example output produced by the DA and DM model. The source sentence contains the following three errors: change “asa" to "as", change "extraorinbary" to "extraordinary", and insert "a" before “rumba”. Failure to correct the first error causes DA to fail to correct "extraorinbary" to "extraordinary" and to fail to insert "a" before “rumba”. However, DM successfully corrects the last two errors even when it fails to correct the first error.

4.5.2 Length and Edit Analysis

In this section, we further analyze the effect of our method on reducing the error accumulation problem, which tends to be more severe with longer sentences or sentences that require more edits. Therefore, we analyze the performance with respect to different lengths of the source sentence and different numbers of edits in the gold annotation in Figure 3.

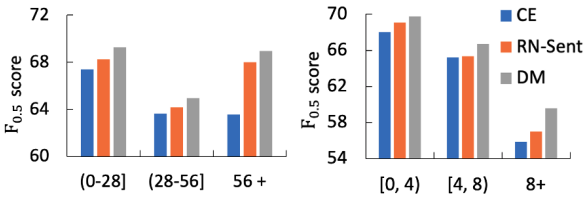

Figure 3: Performance comparison on the CoNLL-2014 test set with respect to different lengths of the source sentence, and different numbers of edits in the gold annotation. *Left:* source sentence length; *Right:* number of edits in the gold annotation.

Our method consistently improves over both CE and RN-Sent in all cases. The improvement is larger when the length of the source sentence is greater than 56, or the number of edits in the gold

reference is greater than 8. This further shows that our method effectively addresses the error accumulation problem and that our improvement is mainly obtained by addressing the exposure bias problem.

4.5.3 Performance Analysis

This section further compares DM's capability of addressing the exposure bias problem against other baseline methods.

We consider the exposure bias problem to be better addressed when the predicted probability for the ground truth token is higher, under the condition that previously predicted tokens are generated by the model itself (i.e., not the ground truth). To evaluate how the exposure bias problem is addressed, we further design an experiment, which is an improved version of Zhang et al. (2019).

Specifically, we first generate noisy sentences by feeding the source sentences from the CoNLL-2014 test set to the model checkpoint before fine-tuning. We then generate the padded noisy sentences using the same procedure described in Section 3.2.1. Subsequently, we use different models (model A and model B in Table 7) to decode the same source sentence by using the padded noisy sentences as the decoder input. Let N be the number of ground truth tokens whose probabilities in the predicted distributions produced by model B are greater than those produced by model A. The win ratio is calculated by dividing N by the total number of tokens in the gold sentences. The results are shown in Table 7.

Win ratio (B \ A)	CE	DM
SS	69.99%	35.26%
RN-Word	73.54%	39.04%
RN-Sent	76.41%	37.64%
DM	89.41%	-

Table 7: The win ratio of model B over model A.

From Table 7, we observe that more than 89% of the target tokens' predicted probabilities of our DM model are greater than the CE model, indicating that our DM method greatly reduces the exposure bias problem. Moreover, compared to the DM model, all the other models are shown to have a win ratio of less than 40%. This shows that our method better addresses the exposure bias problem in GEC.

5 Conclusion

In this paper, we propose a novel data manipulation approach that includes both data augmentation and data reweighting to mitigate the exposure bias problem for seq2seq GEC models. Our core idea is to inject realistic augmented sentences into the decoder input and to automatically sample the augmented sentences. We show the effectiveness of our method which achieves significant improvements on both the CoNLL-2014 and BEA-2019 test sets.

6 Limitations

While our proposed approach is language-independent, we have only tested our approach on the English GEC task.

Acknowledgements

We thank Muhammad Reza Qorib for helpful comments on this paper. This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-014). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, page 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.

Hannan Cao, Wenmian Yang, and Hwee Tou Ng. 2021. Grammatical error correction with contrastive learning in low error density domains. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4867–4874.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. 2017. Learning to learn without gradient descent by gradient descent. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 748–756.

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.

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*, pages 825–835.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1563–1572.

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.

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*, 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*, pages 634–646.

Tsvetomila Mihaylova and André F. T. Martins. 2019. Scheduled sampling for transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356.

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.

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.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, 2016, Conference Track Proceedings*.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 702–707.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372.

Danielle Saunders, Felix Stahlberg, and Bill Byrne. 2020. Using context in neural machine translation training objectives. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7764–7770.

Chenze Shao, Xilin Chen, and Yang Feng. 2018. Greedy search with probabilistic n-gram matching for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4778–4784.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1917–1928.

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.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5937–5947.

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*, pages 5998–6008.

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.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, pages 229–256.

Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Lai Jian-Huang, and Tie-Yan Liu. 2018. Learning to teach with dynamic loss functions. In *Advances in Neural Information Processing Systems*, pages 6467–6478.

Haoran Xu, Hainan Zhang, Yanyan Zou, Hongshen Chen, Zhuoye Ding, and Yanyan Lan. 2021. Adaptive bridge between training and inference for dialogue generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2541–2550.

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.

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*, pages 4334–4343.

A Appendix

A.1 Error Density of RN-Sent

We adopt a similar approach to measure the error density in RN-Sent when training on the W&I training set. Specifically, our DM method only requires one epoch to converge, while RN-Sent requires seven epochs to converge. The average edits for each augmented sentence is 7.82, and the probability of creating augmented sentences is shown in Table 8.

epoch	p
1	0.01
2	0.02
3	0.03
4	0.05
5	0.07
6	0.11
7	0.17

Table 8: Probability of creating augmented sentences for each epoch.

The high edits per sentence generated by RN-Sent confirm that the augmented sentence generated through random noise injection has a bigger gap between actual decoder input and less efficiently addresses the exposure bias problem. Note that the original decay hyper-parameter is set to 5,800, with a maximum 40,000 updates[11]. Since our model is trained for 7 epochs (690 updates), we adjust the decay hyper-parameter to 100 accordingly.

A.2 RE-DP Setting

When using RE-DP to reweight each augmented sentence, we follow the setting in (Lichtarge et al., 2020). Specifically, we treat D_{aug} as the base dataset D^-, and D_{train} as the target dataset D^+. For a trained GEC model θ_{train}, it has been firstly pre-trained on $C4_{200M}$ and then trained on the combination of NUCLE, FCE, and CLang-8 dataset. We first obtain the model θ^- by fine-tuning θ_{train}

[11] https://github.com/ictnlp/OR-NMT#results-and-settings-on-wmt14-english-german-translation-task

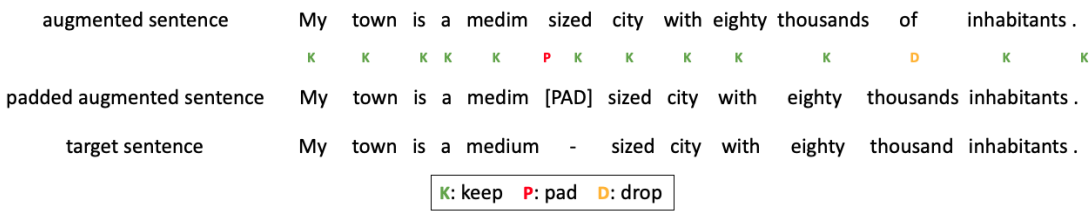

Figure 4: Illustration of the padding mechanism.

on D^- with the following loss function:

$$L_{RE-DP} = \sum_{j=1}^{n} l(x, \bar{y}_j, y; \theta)$$

The model θ^+ is obtained by further fine-tuning θ^- on D^+ using Eq. 2. Based on θ^- and θ^+, we calculate the perplexity of each augmented sentence. The weight $\alpha_{x,j}$ for each augmented sentence is calculated by the 'delta-perplexity-rank' method proposed in (Lichtarge et al., 2020). We then normalize $\alpha_{x,j}$, such that $\sum_{j=1}^{n} \alpha_{x,j} = 1$, and use Eq. 4 to fine-tune θ_{train}.

A.3 BLEU vs Edit Distance

We compare the performance of using BLEU to select sentence-level oracle with the performance of using edit distance to select sentence-level oracle in Table 9.

	CoNLL-2014	BEA-2019
BLEU	65.5	70.0
EDIT	66.3	70.8

Table 9: $F_{0.5}$ score on the CoNLL-2014 and BEA-2019 test sets. BLEU: represents selecting sentence-level oracle with the highest BLEU score. EDIT: represents selecting sentence-level oracle with the smallest edit distance.

A.4 Padding Mechanism

In Figure 4, we illustrate the padding mechanism introduced in Section 3.2.1.

A.5 Dataset Usage

The usage of different datasets for different setups is summarized in Table 10. Specifically, the W&I test set contains 4,477 sentences and the CoNLL-2014 test set contains 1,312 sentences.

A.6 Experimental Details

In this part, we will introduce the software packages we have used and the implementation details.

	CoNLL-2014	BEA-2019
Pre-training	$C4_{200M}$	
Training	NUCLE, FCE, CLang-8	
Fine-tuning	W&I train	W&I train
Validation	CoNLL-2013	W&I dev
Test	CoNLL-2014	W&I test

Table 10: Dataset usage for different setup cases.

Software configuration All models are implemented based on the Fairseq[12] and PyTorch packages. More specifically, we use Python 3.7 and PyTorch 1.7.1.

Implementation details When calculating the reward for the data scorer in Eq. 8, we use all the validation data to calculate the validation gradient.

To balance efficiency and performance, we set the update frequency f in line 3 of Algorithm 1 to 250 and 220 when fine-tuning for CoNLL-2014 and BEA-2019, respectively. The update frequency is selected from the range {250, 300} with a step size of 10 using grid search.

A.7 Hyper-Parameters and Computational Budget

We list the hyper-parameters for pre-training and training in Table 11. The hyper-parameters to fine-tune the model with the DM approach are shown in Table 12 . Specifically, pre-training takes 202 hours to converge, and training takes 0.6 hours to converge. Fine-tuning on the W&I training set takes 0.4 hours to converge.

A.8 Performance on CWEB-S/G and JFLEG Test Sets

We show the performance on the CWEB-S/G and JFLEG test sets in Table 13. When testing on the CWEB-S/G test set, we follow the setting in Cao et al. (2021) to extract the CWEB-train and CWEB-dev data. We use CWEB-train as the fine-tuning set

[12]https://github.com/pytorch/fairseq/tree/9f4256edf60554afbcaadfa114525978c141f2bd

Pretraining	
Configuration	Value
Devices	2 NVIDIA A100 GPU
Model architecture	Transformer ("large" setting)
Optimizer	Adam (β_1 = 0.9, β_2 = 0.98, $\epsilon = 1 \times 10^{-8}$)
Learning rate	3.00×10^{-4}
Learning rate scheduler	Inverse sqrt
Dropout	0.3
Warmup	8000
Number of epochs	10
Best epoch	8
Training	
Configuration	Value
Devices	1 NVIDIA A100 GPU
Learning rate	3.00×10^{-5}
Warmup	4000
Best epoch	1

Table 11: Pre-training and training configuration.

CoNLL-2014 & BEA-2019	
Configuration	Value
Devices	1 NVIDIA A100 GPU
Model architecture	Transformer ("large" setting)
Optimizer	Adam (β_1 = 0.9, β_2 = 0.98, $\epsilon = 1 \times 10^{-8}$)
Learning rate	3.00×10^{-5}
Learning rate scheduler	Inverse sqrt
Warmup	4000
Number of epochs	10
Best epoch	1

Table 12: DM fine-tuning configuration.

and CWEB-dev as the validation set. When testing on the JFLEG test set, we still use the W&I training set as the fine-tuning set and use the JFLEG-dev set as the validation set.

Approach	CWEB-S	CWEB-G	JFLEG
CE	38.2	43.0	62.9
SS	33.8	46.6	63.1
RN-Word	37.3	47.8	62.6
RN-Sent	37.7	47.7	63.0
Ad-Bridge	38.8.	47.5	63.1
DA	42.4*	47.0*	63.2*
DA + RE-DP	40.1	47.8	62.7
DM	42.7*†	49.0*†	63.5*†

Table 13: The $F_{0.5}$ score and GLEU score (in %) on the CWEB-S/G test set and JFLEG test set respectively. Statistically significant improvements ($p < 0.01$) over CE and DA are marked as * and † respectively.